
FMMPART3D

Version 1.0
FORTRAN
Beta release

User's guide

Contents

| | | |
|----------|---|-----------|
| 1 | Release Notes | 1 |
| 2 | The FMMPART3D Subroutine | 2 |
| 2.1 | Subroutine FMMPART3D parameters | 2 |
| 3 | The FMMPARTTARG3D Subroutine | 5 |
| 3.1 | Subroutine FMMPARTTARG3D parameters | 5 |
| 4 | The CLOSE subroutine | 8 |
| 4.1 | Subroutine CLOSE parameters | 8 |
| 5 | A sample driver for FMMPART3D | 11 |
| 5.1 | Sample Run | 14 |
| 6 | A sample driver for FMMPARTTARG3D | 15 |
| 6.1 | Sample Run | 18 |
| 7 | Contact Information | 20 |

1 Release Notes

This is the first release of the codes FMMPART3D and FMMPARTTARG3D.

FMMPART3D rapidly evaluates the potentials and fields due to three-dimensional distributions of N charged particles. In its present form, FMMPART3D computes

$$\phi(\mathbf{x}_i) = \sum_{\substack{j=1 \\ j \neq i}}^N \frac{q_j}{\|\mathbf{x}_i - \mathbf{x}_j\|} + \frac{p_j(n_j \cdot (\mathbf{x}_i - \mathbf{x}_j))}{\|\mathbf{x}_i - \mathbf{x}_j\|^3}$$

for $i = 1, \dots, N$. q_j is the monopole strength and p_j is the dipole strength. n_j is a unit vector pointing in the direction of the dipole orientation.

FMMPARTTARG3D has the additional capability to evaluate the potentials and fields at specified target locations, \mathbf{y}_i . It computes

$$\phi(\mathbf{y}_i) = \sum_{j=1}^N \frac{q_j}{\|\mathbf{y}_i - \mathbf{x}_j\|} + \frac{p_j(n_j \cdot (\mathbf{y}_i - \mathbf{x}_j))}{\|\mathbf{y}_i - \mathbf{x}_j\|^3}$$

for $i = 1, \dots, N_t$, $j = 1, \dots, N$, with N_t the number of target locations and N the number of source locations.

Subsequent versions will also allow for the imposition of periodic boundary conditions for applications in solid state physics, mechanics, and chemistry.

2 The FMMPART3D Subroutine

In this section, we briefly describe the FMMPART3D subroutine. The calling sequence for the double precision code is of the form

```

subroutine fmmpart3d(sources, nsources, monocharge,
1      dipstr, dipvec, pot, potx, poty, potz,
2      work, lenw, iprec, iflagtype, iderivonoff,
3      close, epsclose, ier, inform)

```

2.1 Subroutine FMMPART3D parameters

Input Parameters:

`sources(3,nsources):real *8` : `sources(:,i)` coordinate of *i*th source in 3 dimensional space.

`nsources:integer *4` : number of sources

`monocharge(nsources):real *8` : `monocharge(i)` is the charge of *i*th monopole source

`dipstr(nsources):real *8` : `dipstr(i)` is the strength of *i*th dipole source

`dipvec(3,nsources):real *8` : `dipvec(:,i)` is the orientation unit vector for *i*th dipole

`iprec:integer *4` : user-specified parameter that determines the desired precision. Allowed values are

```

iprec = 0  for least squares errors < 10-3,
iprec = 1  for least squares errors < 10-6.
iprec = 2  for least squares errors < 10-9.
iprec = 3  for least squares errors < 10-12.

```

`iflagtype:integer *4` : user-specified flag denoting whether monopoles, dipoles, or both are present

```

iflagtype = 0  for only monopoles.
iflagtype = 1  for only dipoles.
iflagtype = 2  for monopoles and dipoles.

```

NOTE : Unused arrays do not need to be allocated in full. Thus, if `iflagtype = 0`, `dipstr` can be dimensioned as a scalar and `dipvec` can be dimensioned as `dipvec(3)` - BUT NOT `dipvec(1)`. If `iflagtype = 1`, `monocharge` can be dimensioned as a scalar.

`iderivonoff:integer *4` : user-specified flag denoting whether or not the first derivatives are to be computed

`iderivonoff = 0` means don't compute.
`iderivonoff = 1` means compute derivatives.

`epsfclose:real *8` : In many applications, the interaction between particles is Coulombic or gravitational at large distances, but more complicated in the near field. The parameter `epsfclose` determines the cutoff distance at which a user-provided subroutine takes responsibility for computing the pairwise interaction. More precisely, if the distance between two sources is less than `epsfclose`, then their interaction is handled by the user-provided routine `close` (see below). An important consequence of this parameter is that the quad-tree created by FMMPART halts when the dimensions of a box are comparable to `epsfclose`. This can result in significant performance degradation if many sources are within a distance `epsfclose` of each other. Thus, if pure Coulomb interactions are desired, the user should set `epsfclose = 0`.

`close:external subroutine` : (see the file `close.f`) A user-provided subroutine evaluating the pairwise interaction between particles when they are closer to each other than the distance `epsfclose` (see above). The user must declare the subroutine being used for close encounters as `external` in the calling program. The user must also be sure that the calling sequence of `close` conforms to the example in the `close.f` file.

Workspace:

`work(lenw):real *8` : real work array

`lenw:integer *4` : length of work (must be at least $20 * \text{nsources}$). $70 * \text{nsources}$ will usually be sufficient.

Output Parameters:

`pot(nsources):real *8` : `pot(i)` is the potential at the `i`th source

`potx(nsources), poty(nsources), potz(nsources):real *8` :

`potx(i)` is the x derivative of the potential at the `i`th source

`poty(i)` is the y derivative of the potential at the `i`th source

`potz(i)` is the z derivative of the potential at the `i`th source

NOTE : Unused arrays do not need to be allocated in full. If `iderivonoff = 0`, `potx`, `poty`, and `potz` can be dimensioned as scalars.

`ier:integer*4` Error return codes.

`ier = 0`: Successful completion of code.

`ier = 1`: normal execution, but `epsclse` 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. terminal error

`ier = 5`: invalid choice of `epsclse`, solution not attempted

`ier = 6`: invalid `nsources` value, solution not attempted

`ier = 7`: problem with the license file, solution not attempted.

`inform(5);integer *4` : `inform` array contains information after successful execution of the code

`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

3 The FMMPARTTARG3D Subroutine

In this section, we briefly describe the FMMPARTTARG3D subroutine. The calling sequence for the double precision code is of the form

```

subroutine fmmparttarg3d(sources, nsources, targets, ntargets,
1     monocharge, dipstr, dipvec, pot, potx, poty, potz,
2     pottar, pottarx, pottary, pottarz,
3     work, lenw, iprec, iflagtype, iderivonoff, itargflag,
4     idivideon, close, epsclose, ier, inform)

```

3.1 Subroutine FMMPARTTARG3D parameters

Input Parameters:

`sources(3,nsources):real *8` : `sources(:,i)` coordinate of *i*th source in 3 dimensional space.

`nsources:integer *4` : number of sources

`targets(3,ntargets):real *8` : `targets(:,i)` coordinate of *i*th target in 3 dimensional space.

`ntargets:integer *4` : number of targets

`monocharge(nsources):real *8` : `monocharge(i)` is the charge of *i*th monopole source

`dipstr(nsources):real *8` : `dipstr(i)` is the strength of *i*th dipole source

`dipvec(3,nsources):real *8` : `dipvec(:,i)` is the orientation unit vector for *i*th dipole

`iprec:integer *4` : user-specified parameter that determines the desired precision. Allowed values are

```

iprec = 0  for least squares errors < 10-3,
iprec = 1  for least squares errors < 10-6.
iprec = 2  for least squares errors < 10-9.
iprec = 3  for least squares errors < 10-12.

```

`iflagtype:integer *4` : user-specified flag denoting whether monopoles, dipoles, or both are present

```

iflagtype = 0  for only monopoles.
iflagtype = 1  for only dipoles.
iflagtype = 2  for monopoles and dipoles.

```

NOTE : Unused arrays do not need to be allocated in full. Thus, if `iflagtype = 0`, `dipstr` can be dimensioned as a scalar and `dipvec` can be dimensioned as `dipvec(3)` - BUT NOT `dipvec(1)`. If `iflagtype = 1`, `monocharge` can be dimensioned as a scalar.

`iderivonoff:integer *4` : user-specified flag denoting whether or not the first derivatives are to be computed

`iderivonoff = 0` means don't compute.
`iderivonoff = 1` means compute derivatives.

`itargflag:integer *4` : user-specified flag that indicates if the solution is to be computed at the sources and targets or at the targets only.

`itargflag = 0` just compute solution at targets.
`itargflag = 1` compute solution at targets and sources.

`idivideon:integer *4` : user-specified flag that indicates if the riverdale tree should be formed by dividing on the sources, targets, or both.

`idivideon = 0` means divide on sources.
`idivideon = 1` means divide on targets.
`idivideon = 2` means divide on both sources and targets.

It is possible to save time or memory by properly setting the `idivideon` flag. It's best value can be determined by experimentation, but the following values are reasonable defaults. If `itargflag=0` and `nsources>>ntargets`, set `idivideon=1`. If `itargflag=0` and `nsources<<ntargets`, set `idivideon=0`. If `itargflag=1` and `nsources<<ntargets`, set `idivideon=0`. Otherwise set `idivideon=2`.

`epsclse:real *8` : In many applications, the interaction between particles is Coulombic or gravitational at large distances, but more complicated in the near field. The parameter `epsclse` determines the cutoff distance at which a user-provided subroutine takes responsibility for computing the pairwise interaction. More precisely, if the distance between two sources is less than `epsclse`, then their interaction is handled by the user-provided routine `close` (see below). An important consequence of this parameter is that the quad-tree created by FMMPART halts when the dimensions of a box are comparable to `epsclse`. This can result in significant performance degradation if many sources are within a distance `epsclse` of each other. Thus, if pure Coulomb interactions are desired, the user should set `epsclse = 0`.

close:external subroutine : (see the file close.f) A user-provided subroutine evaluating the pairwise interaction between particles when they are closer to each other than the distance epsclose (see above). The user must declare the subroutine being used for close encounters as external in the calling program. The user must also be sure that the calling sequence of close conforms to the example in the close.f file.

Workspace:

work(lenw):real *8 : real work array

lenw:integer *4 : length of work (must be at least $20 * \text{nsources}$). $70 * \text{nsources}$ will usually be sufficient.

Output Parameters:

pot(nsources):real *8 : pot(i) is the potential at the ith source

potx(nsources), poty(nsources), potz(nsources):real *8 :

potx(i) is the x derivative of the potential at the ith source

poty(i) is the y derivative of the potential at the ith source

potz(i) is the z derivative of the potential at the ith source

pottar(ntargets):real *8 : pottar(i) is the potential at the ith target

pottarx(ntargets), pottary(ntargets), pottarz(ntargets):real *8 :

pottarx(i) is the x derivative of the potential at the ith target

pottary(i) is the y derivative of the potential at the ith target

pottarz(i) is the z derivative of the potential at the ith target

NOTE : Unused arrays do not need to be allocated in full. If `iderivonoff = 0`, `potx`, `poty`, and `potz` can be dimensioned as scalars.

ier:integer*4 Error return codes.

ier = 0: Successful completion of code.

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. terminal error

`ier = 5`: invalid choice of `eps`close, solution not attempted

`ier = 6`: invalid `n`sources value, solution not attempted

`ier = 7`: problem with the license file, solution not attempted.

`inform(5);integer *4` : `inform` array contains information after successful execution of the code

`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

4 The CLOSE subroutine

The CLOSE subroutine must take the form

```
subroutine close(eps, xinfo1, xinfo2, iflag, xout)
```

The CLOSE subroutine calculates the interaction of two particles in the regime where their interactions are not purely Coulombic. Note that the FMM can only accelerate the computation of the Coulombic interaction. The computation of the close interactions via the CLOSE subroutine will not be accelerated. This subroutine must be supplied by the user and customized to his particular problem. Note that while the contents of the subroutine is determined by the user, the calling sequence must be as given above.

4.1 Subroutine CLOSE parameters

Input Parameters:

`eps`: `real *8` : this is the parameter `eps`close defined in the calling sequence of FMMPART3D or FMMPARTTARG3D. It is provided as a convenience to the user.

`xinfo1(8):real *8` : vector containing the information for the first (incoming) point in the interaction. If FMMPARTTARG3D is in use, and if this is a target location, only elements 1 to 3 of the vector are non-empty, giving the target coordinates. Otherwise, the entire vector is filled as follows.

`xinfo1(1)` x position of incoming point
`xinfo1(2)` y position of incoming point
`xinfo1(3)` z position of incoming point
`xinfo1(4)` the monopole charge at the incoming point
`xinfo1(5)` the dipole charge at the incoming point
`xinfo1(6)` x part of dipole vector at incoming point
`xinfo1(7)` y part of dipole vector at incoming point
`xinfo1(8)` z part of dipole vector at incoming point

`xinfo2(8): real *8` : vector containing the information for the second (outgoing) point in the interaction

`xinfo2(1)` x position of outgoing point
`xinfo2(2)` y position of outgoing point
`xinfo2(3)` z position of outgoing point
`xinfo2(4)` the monopole charge at the outgoing point
`xinfo2(5)` the dipole charge at the outgoing point
`xinfo2(6)` x part of dipole vector at outgoing point
`xinfo2(7)` y part of dipole vector at outgoing point
`xinfo2(8)` z part of dipole vector at outgoing point

`iflag(2):integer *4` : the array of all flags that must be specified when calling the `fmm3d` subroutine

`iflag(1)` is `iflagtype`. indicates whether monopoles, dipoles, or both are present.
`iflag(2)` is `iderivonoff`. indicates whether or not to compute the derivatives. 1 yes, 0 no.

Output Parameters:

`xout(4): real *8` : array of output computed using the techniques specified in the subroutine

`xout(1)` will be added to the potential term
`xout(2)` will be added to the x derivative
`xout(3)` will be added to the y derivative
`xout(4)` will be added to the z derivative

For example, to simply set the interaction to zero within the cutoff distance, you can use the following close routine.

```
subroutine close(eps, xinfo1, xinfo2, iflag, xout)
integer *4 iflag(2)
real *8 eps, xinfo1(8), xinfo2(8), xout(4)
c
xout(1) = 0
if( iflag(2).eq.1 ) then
    xout(2) = 0
    xout(3) = 0
    xout(4) = 0
endif
c
return
end
```

5 A sample driver for FMMPART3D

```

implicit real*8 (a-h,o-z)
C
parameter (namax= 130 000)
parameter (iprec=1)
parameter (iflagtype=0)
parameter (iderivonoff=0)
parameter (lenw=110*namax)
parameter (epsclose=0)
real*8 sources(3,namax),monocharge(namax)
real*8 dipstr(1),dipvec(3,1)
external close
real*8 work(lenw)
real*8 pot(namax),potx(1),poty(1),potz(1)
real*8 d2pot(1000)
integer ier,inform(5)
C
C Create a distribution of charges.
C
pi = 4.0*datan(1.0d0)
nc = 200
nh = 300
nsources = nc*nh
if( nsources.gt.namax ) then
  write(6,*)'namax is too small'
  stop
endif
write(6,*)'trial with ',nsources,' monopole particles'
write(6,*)' '

nat = 0
do i = 1, nc
  phi = (i-1)*2*pi/nc
  do j = 1,nh
    nat = nat + 1
    sources(1,nat) = 0.05d0*cos(phi)
    sources(2,nat) = 0.05d0*sin(phi)
    sources(3,nat) = -0.5d0+(j-1)*1.d0/(nh-1)
    monocharge(nat) = sources(3,nat)
  enddo
enddo
enddo

```

```

C
C Calculate potentials and fields by fmmpart3d.
C
    time0 = second()
    call fmmpart3d(sources,nsources,monocharge,
$     dipstr,dipvec,pot,potx,poty,potz,
$     work,lenw,iprec,iflagtype,iderivonoff,
$     close,epsclse,ier,inform)
    time1 = second()
    tfmm = time1-time0

    if( ier.ne.0 ) then
        write(6,*)'fmmpart3d returns ier=',ier
        stop
    endif
C
    imax = min(30,nsources)
    write(6,*)'from fmmpart3d, potential ='
    write(6,1111)(pot(i),i=1,imax)
C
C Calculate the results by direct calculation,
C using the subroutine direci.
C
    salg = 0.0
    stot = 0.0
    time0 = second()
    do 1100 i=1,imax
        call direci(nsources,sources,monocharge,i,d2pot(i))
1100 continue
    time1 = second()
    tdir = time1-time0
C
C Compare the results.
C
    do 1101 i=1,imax
        difftmp = d2pot(i) - pot(i)
        salg = salg + difftmp*difftmp
        stot = stot + d2pot(i)*d2pot(i)
1101 continue
C
C scale time for direct calculation up to nsources.
C

```

```

      tdirscal = (tdir*nsources)/imax
C
      write(6,*)' '
      write(6,*)'from direct calculation, potential ='
      write(6,1111)(d2pot(i),i=1,imax)
      write(6,*)' '
      write(6,*)'total time for fmm3d =',tfmm
      write(6,*)'fmm3d process ',nsources/tfmm,' particles/second'
      write(6,*)'estimated time for direct computation =',tdirscal
      write(6,*)'fmm speedup factor = ',tdirscal/tfmm
      ealg = sqrt(salg/stot)
      write(6,*)'L2 error of fmm3d =',ealg
      write(6,*)'fmm3d used a work space of ',
$      inform(5)/nsources+1,' words per particle'
1111 format(6d13.5)
      stop
      end

C
C this subroutine calculates potentials from a set of
c monopole charges directly. The purpose of this routine is
c to provide a comparison to fmm3d for timing and accuracy.
C
      subroutine direci(nsources,sources,monocharge,i,dpot)
      implicit real*8 (a-h,o-z)
C
      integer nsources,i
      real*8 sources(3,nsources),monocharge(nsources)
      real*8 dpot
C
      dpot = 0.0
C
      do 101 j = 1,nsources
         if (j .ne. i) then
            rx = sources(1,i) - sources(1,j)
            ry = sources(2,i) - sources(2,j)
            rz = sources(3,i) - sources(3,j)
            rr = rx*rx + ry*ry + rz*rz
            rdis = sqrt(rr)
            dpot = dpot + monocharge(j)/rdis
         endif
101 continue

```

```
return
end
```

5.1 Sample Run

trial with 60000 monopole particles

from fmm_{part3d}, potential =

| | | | | | |
|-------------|-------------|-------------|-------------|-------------|-------------|
| -.55490D+05 | -.57888D+05 | -.59824D+05 | -.61483D+05 | -.62942D+05 | -.64243D+05 |
| -.65413D+05 | -.66469D+05 | -.67426D+05 | -.68295D+05 | -.69084D+05 | -.69800D+05 |
| -.70448D+05 | -.71035D+05 | -.71563D+05 | -.72037D+05 | -.72461D+05 | -.72837D+05 |
| -.73167D+05 | -.73455D+05 | -.73703D+05 | -.73912D+05 | -.74085D+05 | -.74223D+05 |
| -.74328D+05 | -.74402D+05 | -.74446D+05 | -.74461D+05 | -.74448D+05 | -.74409D+05 |

from direct calculation, potential =

| | | | | | |
|-------------|-------------|-------------|-------------|-------------|-------------|
| -.55490D+05 | -.57888D+05 | -.59824D+05 | -.61483D+05 | -.62942D+05 | -.64243D+05 |
| -.65413D+05 | -.66469D+05 | -.67426D+05 | -.68295D+05 | -.69084D+05 | -.69800D+05 |
| -.70448D+05 | -.71035D+05 | -.71563D+05 | -.72037D+05 | -.72461D+05 | -.72837D+05 |
| -.73167D+05 | -.73455D+05 | -.73703D+05 | -.73912D+05 | -.74085D+05 | -.74223D+05 |
| -.74328D+05 | -.74402D+05 | -.74446D+05 | -.74461D+05 | -.74448D+05 | -.74409D+05 |

total time for fmm_{part3d} = 6.94000006

fmm_{part3d} process 8645.53307 particles/second

estimated time for direct computation = 180.000305

fmm speedup factor = 25.9366432

L2 error of fmm_{part3d} = 1.45371812E-07

fmm_{part3d} used a work space of 52 words per particle

6 A sample driver for FMMPARTTARG3D

```

implicit real*8 (a-h,o-z)
C
parameter (namax= 130 000)
parameter (iprec=1)
parameter (iflagtype=0)
parameter (iderivonoff=1)
parameter (lenw=110*namax)
parameter (epsclose=0)
parameter (itargflag=0)
parameter (idivideon=2)
real*8 sources(3,namax),monocharge(namax),targets(3,namax)
real*8 dipstr(namax),dipvec(3,namax)
external close
real*8 work(lenw)
real*8 pot(namax),potx(namax),poty(namax),potz(namax)
real*8 pottar(namax),pottarx(namax),pottary(namax),
$ pottarz(namax)
real*8 d2pot(1000)
integer*4 ier,inform(5)

C
C Create a distribution of charges.
C
pi = 4.0*datan(1.0d0)
nc = 200
nh = 300
nsources = nc*nh
if( nsources.gt.namax ) then
    write(6,*)'namax is too small'
    stop
endif
write(6,*)'trial with ',nsources,' monopole particles'
write(6,*)' '

nat = 0
do i = 1, nc
    phi = (i-1)*2*pi/nc
    do j = 1,nh
        nat = nat + 1
        sources(1,nat) = 0.05d0*cos(phi)
    
```

```

        sources(2,nat) = 0.05d0*sin(phi)
        sources(3,nat) = -0.5d0+(j-1)*1.d0/(nh-1)
        monocharge(nat) = sources(3,nat)
    enddo
enddo
C
C Evaluate potential on a line at x = 0.2, y = 0.2
C
    nz = 10 000
    ntargets = 0
    do i = 1, nz
        ntargets = ntargets + 1
        targets(1,i) = 0.2d0
        targets(2,i) = 0.2d0
        targets(3,i) = (-0.5d0+(i-1)*1.d0/(nz-1))*1.121d0
    enddo
    if( ntargets.gt.namax ) then
        write(6,*)'namax is too small for ntargets'
        stop
    endif
C
C Calculate potentials and fields by fmmparttarg3d.
C
    time0 = second()
    call fmmparttarg3d(sources,nsources,targets,ntargets,
$     monocharge,dipstr,dipvec,pot,potx,poty,potz,
$     pottar, pottarx, pottary, pottarz,
$     work,lenw,iprec,iflagtype,iderivonoff,itargflag,
$     idivideon,close,epsfclose,ier,inform)
    time1 = second()
    tfmm = time1-time0

    if( ier.ne.0 ) then
        write(6,*)'fmmparttarg3d returns ier=',ier
        stop
    endif
C
    imax = min(30,nsources)
    write(6,*)'from fmmparttarg3d, potential at targets ='
    write(6,1111)(pottar(i),i=1,imax)
C
C Calculate the results by direct calculation,

```

```

C   using the subroutine directargi.
C
      salg = 0.0
      stot = 0.0
      time0 = second()
      do 1100 i=1,imax
          call directargi(nsources,sources,targets,
$   monocharge,i,d2pot(i))
1100  continue
      time1 = second()
      tdir = time1-time0
C
C   Compare the results.
C
      do 1101 i=1,imax
          difftmp = d2pot(i) - pottar(i)
          salg = salg + difftmp*difftmp
          stot = stot + d2pot(i)*d2pot(i)
1101  continue
C
C   scale time for direct calculation up to nsources.
C
      tdirscal = (tdir*nsources)/imax
C
      write(6,*)' '
      write(6,*)'from direct calculation, potential ='
      write(6,1111)(d2pot(i),i=1,imax)
      write(6,*)' '
      write(6,*)'total time for fmm part3d =',tfmm
      write(6,*)'fmm parttarg3d process ',nsources/tfmm,
$   ' particles/second'
      write(6,*)'estimated time for direct computation =',tdirscal
      write(6,*)'fmm speedup factor = ',tdirscal/tfmm
      ealg = sqrt(salg/stot)
      write(6,*)'L2 error of fmm parttarg3d =',ealg
      write(6,*)'fmm parttarg3d used a work space of ',
$   inform(5)/nsources+1,' words per particle'
1111  format(6d13.5)
      stop
      end
C

```

```

C   this subroutine calculates potentials from a set of
C   monopole charges directly.  The purpose of this routine is
C   to provide a comparison to fmmparttarg3d for timing and
C   accuracy.
C
C   subroutine directargi(nsources,sources,targets,
$   monocharge,i,dpot)
C   implicit real*8 (a-h,o-z)
C
C   integer nsources,i
C   real*8 sources(3,nsources),monocharge(nsources)
C   real*8 dpot, targets(3,1)
C
C   dpot = 0.0
C
C   do 101 j = 1,nsources
C       rx = sources(1,j) - targets(1,i)
C       ry = sources(2,j) - targets(2,i)
C       rz = sources(3,j) - targets(3,i)
C       rr = rx*rx + ry*ry + rz*rz
C       rdis = sqrt(rr)
C       dpot = dpot + monocharge(j)/rdis
101 continue
C   return
C   end

```

6.1 Sample Run

trial with 60000 monopole particles

from fmmparttarg3d, potential at targets =

| | | | | | |
|-------------|-------------|-------------|-------------|-------------|-------------|
| -.12819D+05 | -.12821D+05 | -.12824D+05 | -.12826D+05 | -.12829D+05 | -.12831D+05 |
| -.12834D+05 | -.12836D+05 | -.12839D+05 | -.12841D+05 | -.12844D+05 | -.12847D+05 |
| -.12849D+05 | -.12852D+05 | -.12854D+05 | -.12857D+05 | -.12859D+05 | -.12862D+05 |
| -.12864D+05 | -.12867D+05 | -.12869D+05 | -.12872D+05 | -.12874D+05 | -.12877D+05 |
| -.12879D+05 | -.12882D+05 | -.12884D+05 | -.12887D+05 | -.12889D+05 | -.12892D+05 |

from direct calculation, potential =

| | | | | | |
|-------------|-------------|-------------|-------------|-------------|-------------|
| -.12819D+05 | -.12821D+05 | -.12824D+05 | -.12826D+05 | -.12829D+05 | -.12831D+05 |
| -.12834D+05 | -.12836D+05 | -.12839D+05 | -.12841D+05 | -.12844D+05 | -.12847D+05 |
| -.12849D+05 | -.12852D+05 | -.12854D+05 | -.12857D+05 | -.12859D+05 | -.12862D+05 |
| -.12864D+05 | -.12867D+05 | -.12869D+05 | -.12872D+05 | -.12874D+05 | -.12877D+05 |

-.12879D+05 -.12882D+05 -.12884D+05 -.12887D+05 -.12889D+05 -.12892D+05

total time for fmm3d = 2.80999994
fmm3d process 21352.3136 particles/second
estimated time for direct computation = 140.000343
fmm speedup factor = 49.8221873
L2 error of fmm3d = 1.25707331E-06
fmm3d used a work space of 49 words per particle

7 Contact Information

| | |
|----------------------------------|--|
| on the web | www.madmaxoptics.com |
| email support | support@madmaxoptics.com |
| product information and ordering | sales@madmaxoptics.com |
| general information | info@madmaxoptics.com |
| phone | 203-248-1338 |
| fax | 203-248-1476 |
| mail | MadMax Optics Inc. 3035 Whitney Ave. Hamden CT 06518 |

Correspondence concerning FMMPART3D should be sent to the email support address given above.

References

- [1] H. Cheng, L. Greengard and V. Rokhlin, "A Fast Adaptive Multipole Algorithm in Three Dimensions," *J. Comput. Phys.*, **155** pp.468-498, 1999.
- [2] L. Greengard and V. Rokhlin, "A New Version of the Fast Multipole Method for the Laplace Equation in Three Dimensions," *Acta Numerica*, pp. 229-269, 1997.