Supporting Information

Active Site Pressurization: A New Tool for Structure-Guided Drug Design

Ian M. Withers, Michael P. Mazanetz, Wang Hao, Peter M. Fischer, and Charles A. Laughton

1 Introduction

This supporting information provides details of how to implement the ASP methodology within AMBER7. To successfully add ASP a user must have;

- A fully patched and licensed version of AMBER7
- A copy of the ASP.patch file

The ASP.patch file contains updates to the *sander* module of AMBER7 which results in the creation of two new executables, `sander_grid` and `sander_fluid`, which implement the grid and fluid versions of ASP respectively. The entire contents of the ASP.patch file are listed here (in section 5) or may be obtained from the authors, by sending a request by e-mail to `asp@holmes.cancres.nottingham.ac.uk`.

The resulting executables have been tested on a variety hardware platforms running a variety of flavours of UNIX. However, the authors make no warranty, expressed or implied, that the program will function without error, or in any particular hardware environment, or so as to generate any particular function or result, or as to the condition of the program, its merchantability, or its fitness for a particular purpose.

2 Installation

To modify AMBER7 to include the ASP methodology;

1. Set the `$AMBERHOME` environment variable to point to where the AMBER7 tree resides on your machine. For example

   ```
   Using csh, tcsh, etc:  setenv AMBERHOME /usr/local/amber7

   Using bash, sh, etc:   export AMBERHOME=/usr/local/amber7
   ```

2. Create two new directories in the `$AMBERHOME/src` directory which contain fully patched copies of sander;

   ```
   cp -r $AMBERHOME/src/sander $AMBERHOME/src/sander_grid
   cp -r $AMBERHOME/src/sander $AMBERHOME/src/sander_fluid
   ```

3. Save the patch script to create the source code for `sander_grid` and `sander_fluid` in `$AMBERHOME/src`, and apply the patch;

   ```
   cd $AMBERHOME/src
   patch -p0 -N -r ASP_rejects < ASP.patch
   ```

4. Compile and install the modified versions of sander in the usual manner for installing AMBER, in either serial or parallel as described in the installation section of the AMBER7 User Manual.

   ```
   make install
   ```

3 Sander_grid

`sander_grid` is a modified version of the *sander* module of Amber. The purpose of `sander_grid` is to perform energy minimisation and molecular dynamics simulations upon systems containing LJ grid particles, defined by atom type `LJ` and residue type `LJP`. The LJ particles interact with the rest of the system depending upon their status;

1. The particle is *on* and interacts with all other atoms in the system as usual.
2. The particle is *off* and does not interact with any other atoms in the system.
3. The particle is *receptive* and does not interact with any other atoms in the system when force and energy calculations are used to evolve the system. Forces acting upon the *receptive* particles by the system are however accumulated and output.

The status of LJ particles during a molecular dynamics run do not change. To turn *receptive* particles particles *on* the dynamics must be stopped and additional software used to change the particles status.

`sander_grid` can also be used to automatically fill the an active site cavity. To achieve this any *receptive* particles defined in the input file are turned *on* iteratively and the energy of the system is recalculated. If the energy change is less than a user defined value then the particle is flagged as being *on* and any of the six nearest neighbouring grid points which have not yet been considered to be *on* are flagged as *receptive*. If the energy change is greater than the the user defined value (for example, when the particle overlaps another atom) then the particle is flagged as being *off*. This processes is repeated until all grid particles within the cavity have been considered.

3.1 File usage

Usage: `sander_grid` `[-help] [-O] -i mdin -o mdout -p prmtop -c inpcrd -r restrt -gi gridin -go gridout [-ref refc -x mdcrd -v mdvel -e mden -idip inpdip -rdip rstdip -mdip mddip -inf mdinfo -radii radii]`

In addition to the usual files, `sander_grid` also uses the following files;

file	unit	in/out	purpose
`gridin`	17	input	initial state of each grid particle and accumulated forces for each grid particle.
`gridout`	18	output	final state of each grid particle and accumulated forces for each grid particle. This file is written every time the `restrt` file is written.

3.2 Input variables

The LJ particles are held fixed during the simulation using belly. Therefore `ibelly = 1` and all *moving* atoms within the system must be specified in Group format at the end of all other input from file `mdin`.

In addition to all the usual input variables available in `sander`, `sander_grid` has the following additional variables;

`IMIN` Flag to run minimisation.
In addition to the usual options, option 2 has been added to automatically fill the an active site cavity with LJ particles.

- `= 0` No minimisation (only do molecular dynamics; default).
- `= 1` Perform minimisation (and no molecular dynamics).
- `= 2` Automatically fill the active cavity with LJ particles.
- `= 5` Read in a trajectory for analysis.

`GTOL` Change in energy (in kcal mol^{-1}) below which LJ particles will be turned *on* while filling the active site cavity. Default is 0.0.

3.3 Grid File Format

```
FORMAT(4i6,f8.3)   NGRID,NX,NY,NZ,SPACE

  NGRID   :  number of LJ particles
  NX      :  size of the grid along the x axis
  NY      :  size of the grid along the y axis
  NZ      :  size of the grid along the z axis
  SPACE   :  spacing between grid particles

FORMAT(40i2)   (GSTAT(i), i=1,NGRID)

  GSTAT   :  state of grid particle
              = 0, particle is off
              = 1, particle is on
              = 2, particle is imaginary

FORMAT(6e15.7)   (GFORCE(i) ,i=1,3*NGRID)

  GFORCE  :  force acting on grid particles
```

4 Sander_fluid

`sander_fluid` is a modified version of the *sander* module of Amber. The purpose of `sander_fluid` is to perform energy minimisation and molecular dynamics simulations upon systems containing LJ fluid particles, defined by atom type `LJ` and residue type `LJP`. The LJ particles interact with the rest of the system depending upon their status;

1. The particle is *on* and interacts with all other atoms in the system as usual.
2. The particle is *off* and does not interact with any other atoms in the system.
3. The particle is being *grown* in and interacts with the rest of the system via a scaled interaction energy.

LJ fluid particles may only be inserted and grown during a molecular dynamics run. During minimisation the status of all the LJ fluid particles remains fixed.

4.1 File usage

Usage: `sander_fluid [-help] [-O] -i mdin -o mdout -p prmtop -c inpcrd -r restrt -fi fluidin -fo fluidout [-ref refc -x mdcrd -v mdvel -e mden -idip inpdip -rdip rstdip -mdip mddip -inf mdinfo -radii radii -finfo fluidinfo]`

In addition to the usual files, `sander_fluid` also uses the following files;

file	unit	in/out	purpose
`fluidin`	17	input	initial state of fluid particles.
`fluidout`	18	output	final state of fluid particles. This file is written every time the `restrt` file is written.
`fluidinfo`	21	output	user readable LJ particle addition (`ADD`), failure to add (`NADD`), removal (`REM`) and growing (`GROW`) information.

4.2 Input variables

In addition to all the usual input variables available in `sander`, `sander_fluid` has the following additional variables.

`AP` Flag for particle insertion/growth during the MD run.

- `= 0` No particles are added or grown (default).
- `= 1` Particles are added and grown during the MD run.

`NTA` Every `NTA` steps the value of n is reduced by 1.
When $n = 0$ attempts are made to insert a new fluid particle.
Default is 100.

`INCGAM` Value of γ_0. Default is 2.0

`GAMMIN` Value of γ when attempting to insert a new fluid particle.
Default is $2.0^{-19} = 1.90735 \times 10^{-6}$.

The default values of `NTA`, `INCGAM` and `GAMMIN` correspond to a new particle being inserted every 2000 steps.

`GTOL` Change in energy (in kcal mol^{-1}) below which LJ particles will successfully be inserted into the system. Default is 0.0.

`MAXINS` Maximum number of unsuccessful particle insertions before the simulation will terminate. Default is 100.

`GDIA` Cut-off distance (in Å) beyond which fluid particles are considered to have left the fluid. Default is 2.4.

4.3 Fluid File Format

```
FORMAT(i6)    NFLUID

  NFLUID  :  number of LJ particles

FORMAT(40i2)    (GSTAT(i), i=1,NFLUID)

  GSTAT   :  state of grid particle
              = 0, particle is off
              = 1, particle is on
              = 2, particle is being grown

FORMAT(e15.7)    GAM

  GAM     :  scaling factor for particle currently being grown
```

5 ASP.patch

```
--- sander_grid/Makefile
+++ sander_grid/Makefile
@@ -43,7 +43,7 @@
  ddotp.f random.f lsqfit.f amopen.f \
  debug.f ew_recip_reg.f ew_handle_dips.f ew_dipole_recip.f \
  mexit.f egb.f new_time.f extra_pts.f thermo_int.f \
-reorderwat.f matinv.f decomp.f
+ reorderwat.f matinv.f decomp.f fillgrid.f

 OBJ=  sander.o cshf.o noecalc.o noeread.o caldis.o \
       calrate.o dinten.o drates.o indexn.o kmat.o pearsn.o \
@@ -64,39 +64,14 @@
       ddotp.o random.o lsqfit.o amopen.o \
       debug.o ew_recip_reg.o ew_handle_dips.o ew_dipole_recip.o \
       mexit.o egb.o new_time.o extra_pts.o thermo_int.o \
-      reorderwat.o matinv.o decomp.o
+      reorderwat.o matinv.o decomp.o fillgrid.o

-LESOBJ=  sander.LES.o cshf.o noecalc.o noeread.o caldis.o \
-      calrate.o dinten.o drates.o indexn.o kmat.o pearsn.o \
-      plane.o remarc.o nmrcal.LES.o nmrred.LES.o \
-      restal.o getnat.o nmrnrg.o modwt.LES.o disnrg.o angnrg.o \
-      tornrg.o nmrprt.o nmrgrp.o nmrcms.o nmrcmf.o \
-      impnum.o nmrsht.o at2res.o chklin.o opnmrg.o \
-      printe.o runmin.o ndvprt.o force.LES.o rdparm.LES.o \
-      mdread.LES.o locmem.o runmd.LES.o newvel.o getcor.o \
-      r6ave.LES.o r6drv.o aveint.o degcnt.LES.o corf.o \
-      threeb.o tripl.o nmrrad.o decnvh.o \
-      fastwt.o echoin.o parallel.LES.o jnrg.LES.o \
-      shake.o ene.LES.o mdwrit.o minrit.o \
-      set.o setmm.o dynlib.LES.o mdfil.o nmlsrc.o \
-      ew_force.LES.o ew_setup.o ew_box.o ew_bspline.o\
-      ew_fft.o ew_direct.LES.o ew_recip.LES.o pcshift.o align.o \
-      rstack.o istack.o grdmax.o rfree.o rgroup.o \
-      ddotp.o random.o lsqfit.o amopen.o \
-      debug.o ew_recip_reg.o ew_handle_dips.o ew_dipole_recip.o \
-      mexit.o egb.o new_time.o extra_pts.LES.o \
-      thermo_int.o reorderwat.o matinv.o decomp.o
+all::    sander$(SFX)

-all::    sander$(SFX) sander.LES$(SFX)
-
 sander$(SFX): $(OBJ) syslib nxtsec lapack blas
  SYSLIB=`../sysdir lib` ; ../Compile LOAD -o sander$(SFX) \
```

```
  $(OBJ) ../lapack/lapack.a ../blas/blas.a ../lib/nxtsec.o $$SYSLIB;

-sander.LES$(SFX): $(LESOBJ) syslib nxtsec lapack blas
-SYSLIB=`../sysdir lib` ; ../Compile LOAD -o sander.LES$(SFX) \
-$(LESOBJ) ../lapack/lapack.a  ../blas/blas.a ../lib/nxtsec.o $$SYSLIB;
-
 bigsource:
  SYSDIR=`../sysdir dir`  ;  cd $$SYSDIR ; make sys.f
  SYSSRC=`../sysdir src` ;  ../Compile CPPONLY -DDPREC \
@@ -172,6 +147,7 @@
 egb.o: md.h
 egb.o: parms.h
 egb.o: def_time.h
+egb.o: grid.h
 egb.o: gbsa.h
 ene.o: dprec.h
 ene.LES.o: dprec.h
@@ -251,6 +227,8 @@
 ew_direct.LES.o: les.h
 ew_direct.o: ew_nbe.h
 ew_direct.LES.o: ew_nbe.h
+ew_direct.o: grid.h
+ew_direct.LES.o: grid.h
 ew_direct.o: ew_directp.h
 ew_direct.LES.o: ew_directp.h
 ew_direct.o: ew_directe.h
@@ -374,6 +352,20 @@
 fastwt.o: vector.h
 fastwt.o: parallel.h
 fastwt.o: md.h
+fillgrid.o: dprec.h
+fillgrid.o: files.h
+fillgrid.o: md.h
+fillgrid.o: box.h
+fillgrid.o: nmr.h
+fillgrid.o: memory.h
+fillgrid.o: grid.h
+fillgrid.o: extra.h
+fillgrid.o: ew_cntrl.h
+fillgrid.o: ew_mpole.h
+fillgrid.o: def_time.h
+fillgrid.o: ew_unitcell.h
+fillgrid.o: extra_pts.h
+fillgrid.o: parallel.h
 force.o: dprec.h
 force.LES.o: dprec.h
```

```
 force.o: vector.h
@@ -412,11 +404,15 @@
 force.LES.o: extra.h
 force.o: tgtmd.h
 force.LES.o: tgtmd.h
+force.o: grid.h
+force.LES.o: grid.h
 force.o: les.h
 force.LES.o: les.h
 getcor.o: dprec.h
 getcor.o: files.h
 getcor.o: ew_cntrl.h
+getcor.o: grid.h
+getcor.o: parallel.h
 getnat.o: dprec.h
 grdmax.o: dprec.h
 impnum.o: dprec.h
@@ -479,6 +475,8 @@
 mdread.LES.o: sizes.h
 mdread.o: tgtmd.h
 mdread.LES.o: tgtmd.h
+mdread.o: grid.h
+mdread.LES.o: grid.h
 mdread.o: les.h
 mdread.LES.o: les.h
 mdread.o: extra_pts.h
@@ -496,11 +494,13 @@
 mdwrit.o: dprec.h
 mdwrit.o: files.h
 mdwrit.o: ew_unitcell.h
+mdwrit.o: grid.h
 minrit.o: dprec.h
 minrit.o: md.h
 minrit.o: files.h
 minrit.o: box.h
 minrit.o: ew_unitcell.h
+minrit.o: grid.h
 modwt.o: dprec.h
 modwt.o: nmr.h
 ndvprt.o: dprec.h
@@ -604,6 +604,8 @@
 parallel.LES.o: new_time.h
 parallel.o: tgtmd.h
 parallel.LES.o: tgtmd.h
+parallel.o: grid.h
+parallel.LES.o: grid.h
```

```
 pcshift.o: dprec.h
 pcshift.o: vector.h
 pcshift.o: nmr.h
@@ -653,6 +655,8 @@
 rdparm.LES.o: sizes.h
 rdparm.o: istack.h
 rdparm.LES.o: istack.h
+rdparm.o: grid.h
+rdparm.LES.o: grid.h
 remarc.o: dprec.h
 remarc.o: vector.h
 remarc.o: nmr.h
@@ -702,6 +706,7 @@
 sander.o: parms.h
 sander.o: extra.h
 sander.o: tgtmd.h
+sander.o: grid.h
 sander.o: les.h
 sander.o: parallel.h
 sander.o: ew_parallel.h
@@ -728,6 +733,7 @@
 set.o: parallel.h
 set.o: extra.h
 set.o: nmr.h
+set.o: grid.h
 setmm.o: dprec.h
 setmm.o: memory.h
 setmm.o: box.h
@@ -744,6 +750,9 @@
 tripl.o: dprec.h
 tripl.o: pol.h
 tripl.o: box.h
+yammpnb.o: dprec.h
+yammpnb.o: parallel.h
+yammpnb.o: md.h

 #-----------LIBS

@@ -922,10 +931,10 @@

 #-----------

-install: sander$(SFX) sander.LES$(SFX)
-/bin/mv sander$(SFX) sander.LES$(SFX) ../../exe
+install: sander$(SFX)
+ /bin/mv sander$(SFX) ../../exe/sander_grid
```

```
 clean:
-../Clean sander$(SFX) sander.LES$(SFX)
+ ../Clean sander$(SFX)

 # DO NOT DELETE
--- sander_grid/egb.f
+++ sander_grid/egb.f
@@ -3,7 +3,7 @@
      $     epol,eelt,evdw,esurf,dvdl,vdwrad,ineighbor,p1,p2,p3,p4,
      $     r2x,rjx,vectmp1,vectmp2,vectmp3,vectmp4,vectmp5,
      $     vectmp6,vectmp7,vectmp8,jj,skipv,
-     $     k_vals,j_vals,psi,rbmax,rbmin,rbave,rbfluct)
+     $     k_vals,j_vals,psi,rbmax,rbmin,rbave,rbfluct,gstat,type)
 C
 C***********************************************************************
 C                               AMBER                                  **
@@ -104,6 +104,7 @@
 #include "md.h"
 #include "parms.h"
 #include "def_time.h"
+#include "grid.h"
 c
       logical onstep,onstepi
       logical skipv(*)
@@ -128,6 +129,7 @@
      $       dedx,dedy,dedz,qi2h,temp7,daix,daiy,daiz,dij2i,datmp,dij3i,
      $       qid2h,si2,rj1i,dvdl,reff_i,psi,thi,thi2,
      $       dfac,beta,gamma,gamfac,fgbiorig
+      _REAL_ gstat(*),type
 #ifdef ACE
       _REAL_ sigma,omega,vtilde,mu4,r3,r4
 #endif
@@ -141,6 +143,7 @@
      $        iexcl,iaci,jexcl,jexcl_last,jjv,ic
       integer icase, j1, j2
+      integer min
 c
       dimension x(*),xx(*),ix(*),f(*),rborn(*),charge(*),iac(*),
      $          ico(*),numex(*),natex(*),fs(*),reff(*),sumdeijda(*),
@@ -166,7 +169,7 @@
       intdieli = 1.d0/intdiel
       dielfac = intdieli - extdieli
       maxi = natom
-      if(natbel.gt.0) maxi = natbel
```

```
+      if(type.eq.1.0.AND.natbel.gt.0) maxi = natbel
 c
       if( igb.eq.3 .or. igb.eq.4 ) then
         ! Special treatment for Jayaram et al. (M)GB models
@@ -199,6 +202,7 @@
 #else
       do i=1,natom
 #endif
+      if(gstat(i).eq.type) then
         xi = x(3*i-2)
         yi = x(3*i-1)
         zi = x(3*i)
@@ -215,8 +219,14 @@
 c       its own intrinsic radius will be added in
 c
         icount = 0
-        do 25 j=i+1,natom
+        if(type.eq.1.0) then
+           min = i+1
+        else
+           min = 1
+        endif
+        do 25 j=min,natom
 c
+         if(gstat(j).eq.1.0.AND.i.ne.j) then
           xij = xi - x(3*j-2)
           yij = yi - x(3*j-1)
           zij = zi - x(3*j  )
@@ -226,6 +236,7 @@
           icount = icount + 1
           jj(icount) = j
           r2x(icount) = r2
+         endif
 c
    25   continue
 c
@@ -333,6 +344,7 @@
 c
         reff(i) = reff_i
 c
+        endif
       end do
 c
 #ifdef MPI
@@ -390,7 +402,6 @@
       call timer_stop(TIME_gbrad1)
```

```
 c
    90 continue
-c
 c-------------------------------------------------------------------------------
 c
 c     Step 2: loop over all pairs of atoms, computing the gas-phase
@@ -426,6 +437,7 @@
       iexcl = 1
       do i=1,maxi
 #endif
+      if(gstat(i).eq.type) then
         xi = x(3*i-2)
         yi = x(3*i-1)
         zi = x(3*i  )
@@ -440,7 +452,12 @@
 c
 c         -- check the exclusion list for eel and vdw:
 c
-        do k=i+1,natom
+        if(type.eq.1.0) then
+           min = i+1
+        else
+           min = 1
+        endif
+        do k=min,natom
           skipv(k) = .false.
         enddo
         do jjv=jexcl,jexcl_last
@@ -448,8 +465,14 @@
         enddo
 c
         icount = 0
-        do 150 j=i+1,natom
+        if(type.eq.1.0) then
+           min = i+1
+        else
+           min = 1
+        endif
+        do 150 j=min,natom
 c
+         if(gstat(j).eq.1.0.AND.i.ne.j) then
           xij = xi - x(3*j-2)
           yij = yi - x(3*j-1)
           zij = zi - x(3*j  )
@@ -461,6 +484,7 @@
           jj(icount) = j
```

```
           r2x(icount) = r2
           rjx(icount) = reff(j)
+         endif
 c
   150   continue
 #ifdef MASSLIB
@@ -700,6 +724,7 @@
 #else
         iexcl = iexcl + numex(i)
 #endif
+      endif
       end do
       call timer_stop(TIME_gbfrc)
 c
@@ -752,6 +777,7 @@
         do i=1,maxi
 #endif
 c
+      if(gstat(i).eq.type) then
           qi = charge(i)
           expmkf = exp( -kappa * reff(i) )*extdieli
           dl = intdieli - expmkf
@@ -785,6 +811,8 @@
           do 250 j=1,natom
             if( i.eq.j ) go to 250
 c
+           if(gstat(j).eq.1.0) then
+
             da(3*j-2) = 0.0d0
             da(3*j-1) = 0.0d0
             da(3*j  ) = 0.0d0
@@ -798,6 +826,7 @@
             icount = icount + 1
             jj(icount) = j
             r2x(icount) = r2
+           endif
 c
   250     continue
 #ifdef MASSLIB
@@ -927,6 +956,7 @@
             ineighbor(count)=0
           end if
 c
+          endif
         end do   ! end loop over atom i
 c
```

```
         if ( gbsa.eq.1 ) then
--- sander_grid/ew_direct.f
+++ sander_grid/ew_direct.f
@@ -693,7 +693,7 @@
      $     maxnblst,eelt,evdw,ehb,virial,eedvir,
      $     filter_cut,ee_type,eedmeth,dxdr,
      $     epol,dipole,field,mpoltype,r_stack,i_stack,
-     $     gindexlo,gindexhi)
+     $     gindexlo,gindexhi,gstat,gforce,ngrid)

       implicit none
 #ifdef MPI
@@ -723,6 +723,8 @@
       _REAL_ r_stack(*)

       integer nn
+      integer ngrid
+      _REAL_ gstat(*),gforce(3,*)

 c
 c     ---FLOW CONTROL FLAG (debug and future respa)
@@ -762,7 +764,7 @@
      $             eed_cub,eed_lin,charge,
      $             ntypes,iac,ico,cn1,cn2,asol,bsol,filter_cut,
      $             eelt,evdw,ehb,force,virial,ee_type,eedmeth,dxdr,
-     $             eedvir,r_stack(l_real),i_stack(l_int)
+     $             eedvir,r_stack(l_real),i_stack(l_int),gstat,gforce
      $                    )
              elseif ( mpoltype .eq. 1 )then
               call short_ene_dip(i,xk,yk,zk,ipairs(numpack),ntot,nvdw,
@@ -1676,7 +1678,7 @@
      $        eed_cub,eed_lin,charge,
      $        ntypes,iac,ico,cn1,cn2,asol,bsol,filter_cut,
      $        eelt,evdw,ehb,frc,virial,
-     $        ee_type,eedmeth,dxdr,eedvir,tempre,tempint)
+     $        ee_type,eedmeth,dxdr,eedvir,tempre,tempint,gstat,gforce)
       implicit none
       _REAL_ xk,yk,zk,imagcrds(3,*)
       integer i,numvdw,numtot,bckptr(*)
@@ -1696,6 +1698,7 @@
       _REAL_ comm1
       _REAL_ xktran(3,18)
       _REAL_ e3dx,e4dx
+      _REAL_ gstat(*),gforce(3,*)
       integer mask27
       integer nprefetch
```

```
 #ifdef LES
@@ -1706,6 +1709,7 @@
 #include "parallel.h"
 #include "ew_tran.h"
 #include "ew_cntrl.h"
+#include "grid.h"
       integer itran
 c
 c new variables.
--- sander_grid/ew_directe.h
+++ sander_grid/ew_directe.h
@@ -4,6 +4,7 @@
       do im_new = 1,icount
        j = tempint(im_new)

+        if(gstat(i).ne.0.0.AND.gstat(j).ne.0.0) then
         dfee = tempre(5*im_new-4)
         delx = tempre(5*im_new-3)
         dely = tempre(5*im_new-2)
@@ -20,12 +21,23 @@
           f6 = cn2(ic)*r6
           f12 = cn1(ic)*(r6*r6)
 #endif
-          evdw = evdw + f12 - f6
           df = dfee + (12.d0*f12 - 6.d0*f6)*delr2inv

           dfx = delx*df
           dfy = dely*df
           dfz = delz*df
+c ** If particle i is a test point for a future particle then
+          if(gstat(i).gt.1.0) then
+     gforce(1,i-gstart+1) = gforce(1,i-gstart+1)-dfx
+     gforce(2,i-gstart+1) = gforce(2,i-gstart+1)-dfy
+     gforce(3,i-gstart+1) = gforce(3,i-gstart+1)-dfz
+c ** If particle j is a test point for a future particle then
+          elseif(gstat(j).gt.1.0) then
+            gforce(1,j-gstart+1) = gforce(1,j-gstart+1)+dfx
+     gforce(2,j-gstart+1) = gforce(2,j-gstart+1)+dfy
+     gforce(3,j-gstart+1) = gforce(3,j-gstart+1)+dfz
+   else
+          evdw = evdw + f12 - f6
           vxx = vxx - dfx*delx
           vxy = vxy - dfx*dely
           vxz = vxz - dfx*delz
@@ -38,4 +50,17 @@
           frc(1,j) = frc(1,j) + dfx
```

```
           frc(2,j) = frc(2,j) + dfy
           frc(3,j) = frc(3,j) + dfz
+c ** If i is a grid particle then
+   if(i.ge.gstart.AND.i.lt.gstart+gx*gy*gz) then
+     gforce(1,i-gstart+1) = gforce(1,i-gstart+1)-dfx
+     gforce(2,i-gstart+1) = gforce(2,i-gstart+1)-dfy
+     gforce(3,i-gstart+1) = gforce(3,i-gstart+1)-dfz
+c ** If j is a grid particle then
+   elseif(j.ge.gstart.AND.j.lt.gstart+gx*gy*gz) then
+     gforce(1,j-gstart+1) = gforce(1,j-gstart+1)+dfx
+     gforce(2,j-gstart+1) = gforce(2,j-gstart+1)+dfy
+     gforce(3,j-gstart+1) = gforce(3,j-gstart+1)+dfz
+          endif
+   endif
+         endif
         enddo
--- sander_grid/ew_directe2.h
+++ sander_grid/ew_directe2.h
@@ -9,6 +9,7 @@
       do im_new = 1,icount
        j = tempint(im_new)

+        if(gstat(i).ne.0.0.AND.gstat(j).ne.0.0) then
         dfee = tempre(5*im_new-4)
         delx = tempre(5*im_new-3)
         dely = tempre(5*im_new-2)
@@ -26,7 +27,6 @@
           f10 = bsol(ic)*r10
           f12 = asol(ic)*(r10*delr2inv)
 #  endif
-          ehb = ehb + f12 - f10
           df = dfee + (12.d0*f12 - 10.d0*f10)*delr2inv
 #else
           df = dfee
@@ -34,6 +34,18 @@
           dfx = delx*df
           dfy = dely*df
           dfz = delz*df
+c ** If particle i is a test point for a future particle then
+          if(gstat(i).gt.1.0) then
+     gforce(1,i-gstart+1) = gforce(1,i-gstart+1)-dfx
+     gforce(2,i-gstart+1) = gforce(2,i-gstart+1)-dfy
+     gforce(3,i-gstart+1) = gforce(3,i-gstart+1)-dfz
+c ** If particle j is a test point for a future particle then
+          elseif(gstat(j).gt.1.0) then
+            gforce(1,j-gstart+1) = gforce(1,j-gstart+1)+dfx
```

```
+     gforce(2,j-gstart+1) = gforce(2,j-gstart+1)+dfy
+     gforce(3,j-gstart+1) = gforce(3,j-gstart+1)+dfz
+   else
+          ehb = ehb + f12 - f10
           vxx = vxx - dfx*delx
           vxy = vxy - dfx*dely
           vxz = vxz - dfx*delz
@@ -46,4 +58,17 @@
           frc(1,j) = frc(1,j) + dfx
           frc(2,j) = frc(2,j) + dfy
           frc(3,j) = frc(3,j) + dfz
+c ** If i is a grid particle then
+   if(i.ge.gstart.AND.i.lt.gstart+gx*gy*gz) then
+     gforce(1,i-gstart+1) = gforce(1,i-gstart+1)-dfx
+     gforce(2,i-gstart+1) = gforce(2,i-gstart+1)-dfy
+     gforce(3,i-gstart+1) = gforce(3,i-gstart+1)-dfz
+c ** If j is a grid particle then
+   elseif(j.ge.gstart.AND.j.lt.gstart+gx*gy*gz) then
+     gforce(1,j-gstart+1) = gforce(1,j-gstart+1)+dfx
+     gforce(2,j-gstart+1) = gforce(2,j-gstart+1)+dfy
+     gforce(3,j-gstart+1) = gforce(3,j-gstart+1)+dfz
+          endif
+   endif
+         endif
       end do
--- sander_grid/ew_force.f
+++ sander_grid/ew_force.f
@@ -294,7 +294,7 @@
      $       maxnblst,eed,evdw,ehb,dir_vir,eedvir,
      $       nbfilter,ee_type,eedmeth,dxdr,
      $       epold,X(linddip),X(lfield),mpoltype,r_stack,i_stack,
-     $       gindexlo,gindexhi)
+     $       gindexlo,gindexhi,X(lgst),X(lgfor),ngrid)
 #ifdef MPI
         call MPI_COMM_RANK(MPI_COMM_WORLD,mytaskid,ierr)
         call MPI_COMM_SIZE(MPI_COMM_WORLD,numtasks,ierr)
--- sander_grid/files.h
+++ sander_grid/files.h
@@ -2,12 +2,12 @@
       character*80 mdin, mdout, inpcrd, parm, restrt,
      +             refc, mdvel, mden, mdcrd, mdinfo, nmr, mincor,
     +             vecs, radii, freqe,redir(8),
-     +             rstdip,mddip,inpdip
+     +             rstdip,mddip,inpdip,grid,grido
       character owrite
       common /files/ mdin, mdout, inpcrd, parm, restrt,
```

```
      +                refc, mdvel, mden, mdcrd, mdinfo, nmr, mincor,
      +                vecs, radii, freqe, owrite,
-     +             rstdip,mddip,inpdip
+     +             rstdip,mddip,inpdip,grid,gridо

       integer       ITITL(20),ITITL1(20)
       COMMON/RUNHED/ITITL,    ITITL1
--- sander_grid/fillgrid.f
+++ sander_grid/fillgrid.f
@@ -0,0 +1,193 @@
+#include "dprec.h"
+      SUBROUTINE fillgrid(xx,ix,ih,ipairs,X,WINV,amass,F,V,VOLD,XR,XC
+     $     ,CONP,SKIP,NSP,TMA,erstop,r_stack,i_stack,gstat,gforce)
+
+      implicit none
+
+      integer   ipairs(*)
+      _REAL_ xx(*)
+      integer   ix(*), ih(*)
+      _REAL_ r_stack(*)
+      integer i_stack(*)
+#include "files.h"
+#include "md.h"
+#include "box.h"
+#include "nmr.h"
+#include "memory.h"
+#include "grid.h"
+#include "extra.h"
+#include "ew_cntrl.h"
+#include "ew_mpole.h"
+#include "def_time.h"
+#include "ew_unitcell.h"
+#include "extra_pts.h"
+#include "parallel.h"
+      logical do_list_update
+      LOGICAL SKIP(*),erstop
+      _REAL_ X(*),WINV(*),amass(*),F(*),V(*),VOLD(*),
+     $     XR(*),XC(*),CONP(*)
+      _REAL_ VIR(4),ENE(30)
+      _REAL_ TMA(*)
+      integer NSP(*)
+      _REAL_ gstat(*)
+      _REAL_ gforce(3,*)
+
+      integer nix(6),niy(6),niz(6),test(ngrid)
+      integer i,j,k,counttry,countadd,point,pnx,pny,pnz,ixx,iyy,izz,io
```

```
+      real refene
+      character*80 sys
+      logical xp,xn,yp,yn,zp,zn
+
+      data nix/ -1, 1, 0, 0, 0, 0/
+      data niy/  0, 0,-1, 1, 0, 0/
+      data niz/  0, 0, 0, 0,-1, 1/
+
+c     ** Wipe test point array
+      do i = 1,ngrid
+         test(i) = 0
+      enddo
+      point = 0
+      xp = .false.
+      xn = .false.
+      yp = .false.
+      yn = .false.
+      zp = .false.
+      zn = .false.
+c     ** Add initial grid points to test array
+      do i = gstart,gstart+ngrid-1
+         if(gstat(i).eq.2.0) then
+            point = point + 1
+            test(point) = i
+            gstat(i) = 4.0
+         endif
+      enddo
+c     ** Calculate reference energy without any grid particles
+      CALL FORCE(xx,ix,ih,ipairs,X,F,ENE,VIR,r_stack,i_stack, xx(L96),
+     $     xx(L97), xx(L98), do_list_update)
+      refene = ene(1)
+      if(master) then
+         write(6,*)
+         write(6,'(a,f15.4)') 'Energy of system without grid',refene
+         call amflsh(6)
+      endif
+c     ** Loop over test point
+      counttry = 0
+      countadd = 0
+      do while(point.ne.0)
+         counttry = counttry + 1
+         i = test(point)
+         point = point-1
+c     ** Turn this grid particle on
+         gstat(i) = 1.0
+c     ** Calculate energy with this grid point turned on
```

```
+         CALL FORCE(xx,ix,ih,ipairs,X,F,ENE,VIR,r_stack,i_stack, xx(L96)
+     $        ,xx(L97), xx(L98), do_list_update)
+c     ** Determine grid location of this particle
+         j = i-gstart+1
+         pnx = int(real(j-1)/real(gy*gz))
+         pny = int(real(j-1-pnx*gy*gz)/real(gz))
+         pnz = j-pnx*gy*gz-pny*gz
+         pnx = pnx+1
+         pny = pny+1
+c     ** Write to output file
+         if(master) then
+            write(6,*)
+            write(6,'(a,i4,a,3i4)') 'Attempting to add grid point ',i
+     $           -gstart+1,' at ',pnx,pny,pnz
+            write(6,'(a,f15.4)') 'Energy with this point turned on '
+     $           ,ene(1)
+            write(6,'(a,f15.4)') 'Energy difference                '
+     $           ,ene(1)-refene
+         endif
+c     ** If energy increase is within acceptable levels
+         if(ene(1).le.refene+gtol) then
+            gstat(i) = 3.0
+            countadd = countadd + 1
+c     ** Loop over adjacent grid point
+            do k = 1,6
+               ixx = pnx+nix(k)
+               iyy = pny+niy(k)
+               izz = pnz+niz(k)
+c     ** Check for falling off the grid
+               if(ixx.eq.0) then
+                  xn = .TRUE.
+               elseif(ixx.gt.gx) then
+                  xp = .TRUE.
+               elseif(iyy.eq.0) then
+                  yn = .TRUE.
+               elseif(iyy.gt.gy) then
+                  yp = .TRUE.
+               elseif(izz.eq.0) then
+                  zn = .TRUE.
+               elseif(izz.gt.gz) then
+                  zp = .TRUE.
+               else
+c     ** Determine the grid point
+                  j = (ixx-1)*gy*gz + (iyy-1)*gz + izz + gstart-1
+c     ** If this grid point has not been considered previously
+                  if(gstat(j).eq.0.0) then
```

```
+c     ** Add to check list
+                     point = point + 1
+                     test(point) = j
+                     gstat(j) = 4.0
+                  endif
+               endif
+            enddo
+            if(master) write(6,'(a)') '    Added'
+         else
+            gstat(i) = 2.0
+            if(master) write(6,'(a)') 'Not Added'
+         endif
+         if(master) call amflsh(6)
+      enddo
+c     ** Write final summary
+      if(master) then
+         write(6,*)
+         write(6,'(a,i4,a,i4,a)') 'Added ',countadd
+     $        ,' grid points out of ',counttry,' attempted'
+      endif
+c     ** Turn all selected grid particles on
+      do i = gstart,gstart+ngrid-1
+         if(gstat(i).eq.3.0) gstat(i) = 1.0
+      enddo
+c     ** Calculate energy with selected grid particles turned on
+      CALL FORCE(xx,ix,ih,ipairs,X,F,ENE,VIR,r_stack,i_stack, xx(L96),
+     $     xx(L97), xx(L98), do_list_update)
+      if(master) then
+         write(6,*)
+         write(6,'(a,f15.4)') 'Energy of system with grid turned on'
+     $        ,ene(1)
+      endif
+
+c     ** WARNINGS FOR EDGE OF GRID BEING REACHED
+      if(master) then
+         if(xn) then
+            write(6,'(a)') 'Lower edge of grid reached in x direction'
+            print*,'Lower edge of grid reached in x direction'
+         endif
+         if(xp) then
+            write(6,'(a)') 'Upper edge of grid reached in x direction'
+            print*,'Upper edge of grid reached in x direction'
+         endif
+         if(yn) then
+            write(6,'(a)') 'Lower edge of grid reached in y direction'
+            print*,'Lower edge of grid reached in y direction'
```

```
+         endif
+         if(yp) then
+            write(6,'(a)') 'Upper edge of grid reached in y direction'
+            print*,'Upper edge of grid reached in y direction'
+         endif
+         if(zn) then
+            write(6,'(a)') 'Lower edge of grid reached in z direction'
+            print*,'Lower edge of grid reached in z direction'
+         endif
+         if(zp) then
+            write(6,'(a)') 'Upper edge of grid reached in z direction'
+            print*,'Upper edge of grid reached in z direction'
+         endif
+      endif
+
+
+
+      RETURN
+      END
--- sander_grid/force.f
+++ sander_grid/force.f
@@ -68,6 +68,7 @@
 #include "files.h"
 #include "extra.h"
 #include "tgtmd.h"
+#include "grid.h"
 #ifdef LES
 #include "les.h"
 #endif
@@ -87,11 +88,13 @@
      $       virvsene,eelt,e3bod,epol,esurf
       _REAL_ epolar,aveper,aveind,avetot,emtot,
      $       dipiter,dipole_temp
+      _REAL_ tforce(3*natom)
       integer l_r2x,l_rjx,l_tmp1,l_tmp2,l_tmp3,l_tmp4,l_tmp5,
      $        l_tmp6,l_tmp7,l_tmp8,l_jj,l_skipv
       integer l_kvls,l_jvls,l_psi
       integer l_da,l_sumd
       integer newbalance
+      integer j
       save newbalance
 C
       call timer_start(TIME_force)
@@ -408,6 +411,8 @@
           call get_istack( l_kvls, natom )
           call get_istack( l_jvls, natom )
```

```
         end if
+C     ** First call to egb to calculate energy/force for particles with
+C     ** gstat(i) = 1
         call egb( x,xx,ix,f,rborn,fs,reff,xx(L15),ix(I04),ix(I06),
      $      ix(I08),ix(I10),xx(L190),r_stack(l_sumd),
      $      cut,r_stack(l_da),ntypes,natom,natbel,epol,eelt,evdw,
@@ -418,12 +423,41 @@
      $      r_stack(l_tmp7),r_stack(l_tmp8),i_stack(l_jj),
      $      i_stack(l_skipv),
      $      i_stack(l_kvls),i_stack(l_jvls),r_stack(l_psi),
-     $      xx(L186),xx(L187),xx(L188),xx(L189) )
+     $      xx(L186),xx(L187),xx(L188),xx(L189),XX(lgst),1.0d0 )
         ene(2) = evdw
         ene(3) = eelt
         ene(4) = epol
         ene(23) = esurf
         ene(21) = dvdl
+C     ** Store force in temporary array and zero
+        do i = 1,natom*3
+           tforce(i) = f(i)
+           f(i) = 0.0
+        enddo
+C     ** Second call to egb to calculate energy/force for particles with
+C     ** gstat(i) = 2
+        call egb( x,xx,ix,f,rborn,fs,reff,xx(L15),ix(I04),ix(I06),
+     $      ix(I08),ix(I10),xx(L190),r_stack(l_sumd),
+     $      cut,r_stack(l_da),ntypes,natom,natbel,epol,eelt,evdw,
+     $      esurf,dvdl,xx(L165),ix(I82),xx(L170),xx(L175),
+     $      xx(L180),xx(L185),r_stack(l_r2x),r_stack(l_rjx),
+     $      r_stack(l_tmp1),r_stack(l_tmp2),r_stack(l_tmp3),
+     $      r_stack(l_tmp4),r_stack(l_tmp5),r_stack(l_tmp6),
+     $      r_stack(l_tmp7),r_stack(l_tmp8),i_stack(l_jj),
+     $      i_stack(l_skipv),
+     $      i_stack(l_kvls),i_stack(l_jvls),r_stack(l_psi),
+     $      xx(L186),xx(L187),xx(L188),xx(L189),XX(lgst),2.0d0 )
+C     ** Store forces in gforce array
+        do i = gstart,gstart+ngrid-1
+           j = i-gstart+1
+           XX(lgfor+j*3-3) = XX(lgfor+j*3-3)+f(i*3-2)+tforce(i*3-2)
+           XX(lgfor+j*3-2) = XX(lgfor+j*3-2)+f(i*3-1)+tforce(i*3-1)
+           XX(lgfor+j*3-1) = XX(lgfor+j*3-1)+f(i*3  )+tforce(i*3  )
+        enddo
+C     ** Restore force from temporary array
+        do i = 1,natom*3
+           f(i) = tforce(i)
+        enddo
```

```
         if( gbsa.eq.1 ) then
           call free_istack( l_jvls )
           call free_istack( l_kvls )
@@ -454,7 +488,8 @@
 c     add force, ene, vir, npair, nhb copies from all nodes
 c            also add up newbalance for nonperiodic.
 c
-      call fdist(F,XX(Lfrctmp),ene,vir,npair,nhb,r_stack,newbalance)
+      call fdist(F,XX(Lfrctmp),XX(lgfor),XX(lgfortmp),ene,vir,npair,nhb
+     $     ,r_stack,newbalance)
       call timer_stop(TIME_collfrc)
 c
 c     ========================= END AMBER/MPI ========================
--- sander_grid/gbsa.h
+++ sander_grid/gbsa.h
@@ -8,6 +8,7 @@
 #else
         do i=1,natom
 #endif
+   if(gstat(i).eq.type) then
           if (ineighbor(count).eq.0) then
             count=count+1
           else
@@ -199,6 +200,7 @@
             totsasa = totsasa + Ai
 c
           end if
+          endif
         end do
 c
 c       --- finished looping over i ---
--- sander_grid/getcor.f
+++ sander_grid/getcor.f
@@ -248,3 +248,28 @@
      +       /5x,'A     =',f10.5,'  B    =',f10.5,'  C     =',f10.5,/,
     +       /5x,'ALPHA =',f10.5,'  BETA =',f10.5,'  GAMMA =',f10.5,/)
       END
+
+
+
+      subroutine getgrid(natom,ngrid,gstat,gforce)
+      implicit none
+      integer i,natom,ngrid,stat(ngrid+1)
+      _REAL_ gstat(*),gforce(*)
+#include "grid.h"
+#include "parallel.h"
```

```
+
+c     ** Read grid info
+      read(17,'(40i2)') (stat(i),i=1,ngrid)
+      read(17,'(6e15.7)') (gforce(i),i=1,ngrid*3)
+
+c     ** All particles in the system interact....
+      do i = 1,natom
+         gstat(i) = 1.0
+      enddo
+c     ** Update grid particle info from the file......
+      do i = 1,ngrid
+         gstat(i+gstart-1) = real(stat(i))
+      enddo
+
+      return
+      end
--- sander_grid/grid.h
+++ sander_grid/grid.h
@@ -0,0 +1,3 @@
+       integer       gx,gy,gz,gstart
+ real          gdia,gtol
+ COMMON/BOX/gx,gy,gz,gstart,gdia,gtol
--- sander_grid/locmem.f
+++ sander_grid/locmem.f
@@ -52,7 +52,9 @@
 C     AMASS   ...  LWINV ! ATOMIC MASSES (inverted in rdparm - see Lmass)
 C     XCHRG   ...  Lpol  ! atomic polarizibilities
 C     C       ...  LCRD  ! COORDINATES
+c     c            Lgst  ! Status of grid particle
 C     F       ...  Lforce! FORCE
+c                  Lgfor ! Force applied to grid points
 C     V       ...  Lvel  ! VELOCITY for MD, work space for min
 C     VOLD    ...  Lvel2 ! OLD VELOCITY for MD
 C     XR      ...  L45   ! Coords rel. to COM of each molecule
@@ -201,7 +203,9 @@
         end if
       endif
       call adj_mem_ptr( r_ptr, LCRD, 3*natom + MXVAR )
+      call adj_mem_ptr( r_ptr, LGST, natom )
       call adj_mem_ptr( r_ptr, Lforce, 3*natom + MXVAR + 40 )
+      call adj_mem_ptr( r_ptr, Lgfor, 3*ngrid )
       if (imin.eq.0) then
         call adj_mem_ptr( r_ptr, Lvel,  3*natom + MXVAR )
         call adj_mem_ptr( r_ptr, Lvel2, 3*natom + MXVAR )
@@ -273,6 +277,7 @@
       end if
```

```
 #ifdef MPI
       call adj_mem_ptr( r_ptr, Lfrctmp, 3*natom + 40 )
+      call adj_mem_ptr( r_ptr, Lgfortmp, 3*ngrid )
 #endif

       if (nmropt.ge.2) then
@@ -476,10 +481,11 @@
         MAXPR = 1
       else
         if( numextra.eq. 0 ) then
-          maxpr_float = natom * (cutoffnb + skinnb)**3 / 3.5d0
+          maxpr_float = 10.0*natom * (cutoffnb + skinnb)**3 / 3.5d0
         else   ! need more nonbon storage with extra points
-          maxpr_float = natom * (cutoffnb + skinnb)**3 / 2.5d0
+          maxpr_float = 10.0*natom * (cutoffnb + skinnb)**3 / 2.5d0
         end if
+
 c
 c       --- cap at maximum possible number of pairs:
 c
--- sander_grid/mdfil.f
+++ sander_grid/mdfil.f
@@ -55,6 +55,8 @@
       inpdip = 'inpdip'
       mddip = 'mddip'
       radii = 'radii'
+      grid = 'grid'
+      grido = 'grido'
 c
 c     --- default status of output: new
 c
@@ -142,6 +144,12 @@
             elseif (arg .eq. '-mdip') then
                 iarg = iarg + 1
                 call getarg(iarg,mddip)
+            elseif (arg .eq. '-gi') then
+                iarg = iarg + 1
+                call getarg(iarg,grid)
+            elseif (arg .eq. '-go') then
+                iarg = iarg + 1
+                call getarg(iarg,grido)
 #ifdef MPI
             elseif (arg .eq. '-p4pg') then
                 iarg = iarg+1
@@ -201,5 +209,5 @@
      +   'usage: sander  [-O] -i mdin -o mdout -p prmtop -c inpcrd ',
```

```
      +   '-r restrt',/19x,'[-ref refc -x mdcrd -v mdvel -e mden ',
      +   '-idip inpdip -rdip rstdip -mdip mddip ',
-     +   '-inf mdinfo -radii radii]')
+     +   '-inf mdinfo -radii radii -gi grid_in -go grid_out]')
       end
--- sander_grid/mdread.f
+++ sander_grid/mdread.f
@@ -33,6 +33,7 @@
 #include "def_time.h"
 #include "sizes.h"
 #include "tgtmd.h"
+#include "grid.h"
 #ifdef LES
 #include "les.h"
 #endif
@@ -68,7 +69,7 @@
      *          surften,iwrap,nrespa,nrespai,gamma_ln,extdiel,intdiel,
      *          cut_inner,icfe,clambda,klambda,
      *          rbornstat,lastrst,lastist,itgtmd,tgtrmsd,tgtmdfrc,
-     *          idecomp,temp0les
+     *          idecomp,temp0les,gtol
 C
 C Define default water residue name and the names of water oxygen & hydrogens
 C
@@ -90,7 +91,8 @@
      *              'RESTRT',RESTRT(1:70) , 'REFC'  ,REFC(1:70)  ,
      *              'MDVEL' ,MDVEL(1:70)  , 'MDEN'   ,MDEN(1:70) ,
      *              'MDCRD' ,MDCRD(1:70)  , 'MDINFO' ,MDINFO(1:70),
-     *              'INPDIP', INPDIP(1:70), 'RSTDIP', RSTDIP(1:70)
+     *              'INPDIP', INPDIP(1:70), 'RSTDIP', RSTDIP(1:70),
+     *              'GRID', GRID(1:70), 'GRIDO', GRIDO(1:70)
 C
 C Echo the input file to the user:
 C
@@ -218,6 +220,10 @@
       idecomp = 0
       lastrst = MAX_RSTACK
       lastist = MAX_ISTACK
+c
+c     tollerance for adding grid particles
+c
+      gtol = 0.0
 C
 C     Check to see if "cntrl" namelist has been defined.
 C
@@ -367,7 +373,7 @@
```

```
     +       'Amber 7  SANDER                  Scripps/UCSF 2002',
     +       /10X,55(1H-)/)
  9309 FORMAT(/80(1H-)/'   1.  RESOURCE   USE: ',/80(1H-)/)
- 9700 FORMAT(/,'File Assignments:',/,12('|',A6,': ',A,/))
+ 9700 FORMAT(/,'File Assignments:',/,14('|',A6,': ',A,/))
       end
 c
 c
@@ -414,6 +420,7 @@
 #include "ew_legal.h"
 #include "def_time.h"
 #include "tgtmd.h"
+#include "grid.h"
 #ifdef LES
 #include "les.h"
 #endif
@@ -505,6 +512,12 @@
       write(6,'(5x,3(a,f10.5))') 'scnb    =',scnb,
      .     ', scee    =',scee
 c
+c
+      if( imin.eq.2 ) then
+        write(6,'(/a)') 'Energy tolerance for inserted particles:'
+        write(6,'(5x,1(a,f10.5))') 'gtol    =',gtol
+      endif
+c
       write(6,'(/a)') 'Frozen or restrained atoms:'
       write(6,'(5x,4(a,i8))') 'ibelly  =',ibelly,', ntr     =',ntr

@@ -839,6 +852,20 @@
               x(L175-1+i) = -0.16038d0
               x(L180-1+i) = -0.00015512d0
               x(L185-1+i) = 0.00016453d0
+           else if (atype.eq.'FE') then
+c ** MADE UP BY IMW 11/6/04
+              x(L165-1+i) = 1.80d0 + 1.4d0
+              x(L170-1+i) = 0.070344d0
+              x(L175-1+i) = -0.019015d0
+              x(L180-1+i) = -0.000022009d0
+              x(L185-1+i) = 0.000016875d0
+           else if (atype.eq.'LJ') then
+c ** MADE UP BY IMW 11/6/04
+              x(L165-1+i) = 0.25*gdia + 1.4d0
+              x(L170-1+i) = 1.0d0
+              x(L175-1+i) = 0.0d0
+              x(L180-1+i) = 0.0d0
```

```
+               x(L185-1+i) = 0.0d0
              else
                write( 0,* ) 'bad atom type: ',atype
                call mexit( 6,1 )
--- sander_grid/mdwrit.f
+++ sander_grid/mdwrit.f
@@ -1,6 +1,6 @@
 #include "dprec.h"
       SUBROUTINE MDWRIT(nstep,NRP,NR,NRES,NTXO,NTR,NTB,X,V,XC,
-     +     BOX,IGRAPH,LBRES,IPRES,TT)
+     +     BOX,IGRAPH,LBRES,IPRES,TT,ngrid,gstat,gforce)
 C
 C*****************************************************************************
 C                                AMBER                                       **
@@ -21,8 +21,8 @@
 C*****************************************************************************
 C
       implicit none
-      integer nstep,NRP,NR,NRES,NTXO,NTR,NTB,IGRAPH,LBRES,IPRES
-      _REAL_ X,V,XC,BOX,TT
+      integer nstep,NRP,NR,NRES,NTXO,NTR,NTB,IGRAPH,LBRES,IPRES,ngrid
+      _REAL_ X,V,XC,BOX,TT,gstat,gforce
       character*89 restrt2
       character*12 num
       integer istart,iend
@@ -36,20 +36,27 @@
       if( first ) then
         if (ntxo.eq.0) then
           call amopen(16,restrt,owrite,'U','W')
+          call amopen(18,grido,owrite,'U','W')
         else
           call amopen(16,restrt,owrite,'F','W')
+          call amopen(18,grido,owrite,'F','W')
         endif
         first = .false.
       else
         if (ntxo.eq.0) then
           call amopen(16,restrt,'O','U','W')
+          call amopen(18,grido,'O','U','W')
         else
           call amopen(16,restrt,'O','F','W')
+          call amopen(18,grido,'O','F','W')
         endif
       endif
       CALL MDWRI2(16,NRP,NR,NRES,NTXO,NTR,NTB,X,V,XC,
      +                  BOX,IGRAPH,LBRES,IPRES,TT)
```

```
+      CALL MDWRI3(ngrid,gstat,gforce)
+
       close(16)
+      close(18)
 c
 c     -- consider whether to save secondary restrt;
 c
@@ -124,3 +131,23 @@
  9028 FORMAT(6F12.7)
       RETURN
       END
+
+
+      subroutine mdwri3(ngrid,gstat,gforce)
+      implicit none
+#include "grid.h"
+      integer i,ngrid,stat(ngrid)
+      _REAL_ gstat(*),gforce(*)
+
+c     ** WRITE OUT GRID STUFF
+      do i = 1,ngrid
+         stat(i)= int(gstat(i+gstart-1))
+      enddo
+
+      write(18,'(4i6,f8.3)') ngrid,gx,gy,gz,gdia
+      write(18,'(40i2)')    (stat(i),i=1,ngrid)
+      write(18,'(6e15.7)')   (gforce(i),i=1,ngrid*3)
+      write(18,'(e15.7)') 1.0
+
+      return
+      end
--- sander_grid/memory.h
+++ sander_grid/memory.h
@@ -3,7 +3,7 @@
 c      more variables to the "locmem" style of memory management
 c
 c       BC_MEMORY is the size of the MEMORY common block:
-#define BC_MEMORY 171
+#define BC_MEMORY 175
 c
       integer       NATOM,NRES,NBONH,NBONA,NTHETH,NTHETA,NPHIH,
      +    NPHIA,NNB,NTYPES,NRCP,NCONP,MAXMEM,NWDVAR,MAXNB,NPARM,
@@ -23,7 +23,8 @@
      +    L240,L241,L242,
      +    m02,m04,m06,m08,m10,m12,m14,m16,m18,i01,
      +    IVM01,IVM02,nrealb,nintb,nholb,npairb,lastr,lasti,lasth,
```

```
-     +    lastpr,lastrst,lastist,nbper,ngper,ndper,ifpert,Lpolp
+     +    lastpr,lastrst,lastist,nbper,ngper,ndper,ifpert,Lpolp,
+     +    Ngrid,LGST,LGFOR,lgfortmp

       COMMON/MEMORY/NATOM,NRES,NBONH,NBONA,NTHETH,NTHETA,NPHIH,
      +    NPHIA,NNB,NTYPES,NRCP,NCONP,MAXMEM,NWDVAR,MAXNB,NPARM,
@@ -43,4 +44,5 @@
      +    L240,L241,L242,
      +    m02,m04,m06,m08,m10,m12,m14,m16,m18,i01,
      +    IVM01,IVM02,nrealb,nintb,nholb,npairb,lastr,lasti,lasth,
-     +    lastpr,lastrst,lastist,nbper,ngper,ndper,ifpert,Lpolp
+     +    lastpr,lastrst,lastist,nbper,ngper,ndper,ifpert,Lpolp,
+     +    Ngrid,LGST,LGFOR,lgfortmp
--- sander_grid/minrit.f
+++ sander_grid/minrit.f
@@ -1,5 +1,5 @@
 #include "dprec.h"
-      SUBROUTINE MINRIT(X)
+      SUBROUTINE MINRIT(X,gstat,gforce,ngrid)
 C
 C*************************************************************************
 C                                AMBER                                  **
@@ -27,7 +27,9 @@
 #include "files.h"
 #include "box.h"
 #include "ew_unitcell.h"
-      DIMENSION X(*)
+#include "grid.h"
+      DIMENSION X(*),gstat(*),gforce(*)
+      integer stat(ngrid)
 C
       NR = NRP
       NR3 = 3*NR
@@ -34,8 +36,10 @@

       if (ntxo.le.0) then
            call amopen(16,restrt,owrite,'U','W')
+           call amopen(18,grido,owrite,'U','W')
       else
            call amopen(16,restrt,owrite,'F','W')
+           call amopen(18,grido,owrite,'F','W')
       endif
 C
       IF (NTXO.ne.0) THEN
@@ -53,7 +57,16 @@
           WRITE(16) NR,DUMM
```

```
           WRITE(16) (X(I),I = 1,NR3)
       ENDIF
+c     ** WRITE OUT GRID STUFF
+      do i = 1,ngrid
+         stat(i)= int(gstat(i+gstart-1))
+      enddo
+      write(18,'(4i6,f8.3)') ngrid,gx,gy,gz,gdia
+      write(18,'(40i2)')    (stat(i),i=1,ngrid)
+      write(18,'(6e15.7)')   (0.0,i=1,ngrid*3)
+      write(18,'(e15.7)') 1.0
       close(unit=16)
+      close(unit=18)
    40 FORMAT(20A4)
   220 FORMAT(I5,F10.5)
   221 FORMAT(I6,F10.5)
--- sander_grid/parallel.f
+++ sander_grid/parallel.f
@@ -66,11 +66,18 @@
 #include "istack.h"
 #include "new_time.h"
 #include "tgtmd.h"
+#include "grid.h"
 c
       dimension xx(*),ix(*),ih(*)
 c
 c     Send and receive common blocks from the master node:
 c
+c  grid.h
+c
+      call mpi_bcast(gx,4,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
+      call mpi_bcast(gdia,2,MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ierr)
+      call MPI_BARRIER(MPI_COMM_WORLD,ierr)
+c
 c  files.h:
 c
       call mpi_bcast(NTPR,BC_HULP,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
@@ -268,7 +275,8 @@
       return
       end
 c-----------------------------------------------------------------------
-      subroutine fdist(f,forcetmp,ene,vir,npair,nhb,r_stack,newbalance)
+      subroutine fdist(f,forcetmp,gforce,gforcetmp,ene,vir,npair,nhb
+     $     ,r_stack,newbalance)
 c*************************************************************
 c   for the "original version", (when mpi_orig is set):
 c
```

```
@@ -307,8 +315,8 @@
 c
 c     Parameters:
 c
-      _REAL_ f(*),forcetmp(*),ene(*),vir(*)
-      _REAL_ r_stack(*)
+      _REAL_ f(*),forcetmp(*),ene(*),vir(*),gforce(*)
+      _REAL_ r_stack(*),gforcetmp(*)
       integer l_rletmps,l_rletmpr
       integer npair,nhb,newbalance
 c
@@ -346,7 +354,20 @@
           f(i) = forcetmp(i)
         enddo
       else
-c
+c     ** Collect all grid force data on master node
+         call mpi_allreduce(gforce,gforcetmp,(3*ngrid)
+     $        ,MPI_DOUBLE_PRECISION,MPI_SUM,MPI_COMM_WORLD,ierr)
+c     ** Zero grid force on all except master node
+         if(master) then
+            do i = 1,3*ngrid
+               gforce(i) = gforcetmp(i)
+            enddo
+         else
+            do i = 1,3*ngrid
+               gforce(i) = 0.0
+            enddo
+         endif
+cc
 c       ---Add all copies of virial and energy and put result back on ALL nodes:
 c
         call mpi_allreduce(f(j),forcetmp(j),35,
--- sander_grid/rdparm.f
+++ sander_grid/rdparm.f
@@ -123,6 +123,7 @@
 #include "pol.h"
 #include "sizes.h"
 #include "istack.h"
+#include "grid.h"
 #ifdef LES
 #  include "les.h"
       integer iexcl,numex,k1,j1
@@ -756,6 +757,20 @@
       read(nf,9128) (rub(i),i=1,numang)
 #endif
```

```
 C
+c     ** Find the first grid particle
+      gstart = 0
+      i = 1
+      do while(gstart.eq.0)
+         if(iH(m04+I-1).EQ."LJ  ") gstart = i
+         i = i+1
+      enddo
+c     ** Overwrite born radius of LJ particles with real radius
+      if(igb.ne.0) then
+         do i = gstart,natom
+           x(L97+i-1) = 0.5*gdia
+        enddo
+      endif
+
       RETURN
  9108 FORMAT(20A4)
  9128 FORMAT(5E16.8)
--- sander_grid/runmd.f
+++ sander_grid/runmd.f
@@ -1463,12 +1463,14 @@
                if(ireorderwat.eq.1)then
                   CALL MDWRIT(nstep,NRP,NR,NRES,NTXO,NTR,NTB,
      $                 r_stack(ltmpX),r_stack(ltmpv),
-     $                 XX(Lcrdr),BOX,ih(m04),ih(m02),IX(I02),T)
+     $                 XX(Lcrdr),BOX,ih(m04),ih(m02),IX(I02),T,
+     $                 ngrid,XX(Lgst),XX(Lgfor))
                else
 #endif
                CALL MDWRIT(nstep,NRP,NR,NRES,NTXO,NTR,NTB,
      $              X,V,
-     $              XX(Lcrdr),BOX,ih(m04),ih(m02),IX(I02),T)
+     $              XX(Lcrdr),BOX,ih(m04),ih(m02),IX(I02),T,
+     $              ngrid,XX(Lgst),XX(Lgfor))
 #ifdef ROWAT
                endif
 #endif
@@ -1500,12 +1502,14 @@
                if(ireorderwat.eq.1)then
                   CALL MDWRIT(nstep,NRP,NR,NRES,NTXO,NTR,NTB,
      +                 r_stack(l_temp),r_stack(ltmpV),
-     $                 XX(Lcrdr),BOX,ih(m04),ih(m02),IX(I02),T)
+     $                 XX(Lcrdr),BOX,ih(m04),ih(m02),IX(I02),T,
+     $                 ngrid,XX(Lgst),XX(Lgfor))
                else
 #endif
```

```
                  CALL MDWRIT(nstep,NRP,NR,NRES,NTXO,NTR,NTB,
     +                 r_stack(l_temp),  V,
-    $                 XX(Lcrdr),BOX,ih(m04),ih(m02),IX(IO2),T)
+    $                 XX(Lcrdr),BOX,ih(m04),ih(m02),IX(IO2),T,
+    $                 ngrid,XX(Lgst),XX(Lgfor))
 #ifdef ROWAT
                endif
 #endif
--- sander_grid/sander.f
+++ sander_grid/sander.f
@@ -39,6 +39,7 @@
 #include "parms.h"
 #include "extra.h"
 #include "tgtmd.h"
+#include "grid.h"
 #ifdef LES
 #include "les.h"
 #endif
@@ -187,6 +188,8 @@
          CALL MDREAD1()
          call amopen(8,parm,'O','F','R')
          CALL RDPARM1(8)
+         call amopen(17,grid,'O','F','R')
+         read(17,'(4i6,f8.3)') ngrid,gx,gy,gz,gdia
 c
 c        --- now, we can allocate memory:
 c
@@ -279,6 +282,7 @@
          call timer_start(TIME_rdcrd)
          CALL GETCOR(9,NR,X(LCRD),X(LVEL),X(LFORCE),
     $        NTX,BOX,IREST,T)
+         CALL GETGRID(natom,ngrid,x(lgst),x(lgfor))
          if( igb.eq.0 .and. induced.eq.1 ) call get_dips(X,nr)
 C
 C        ----- SET THE INITIAL VELOCITIES -----
@@ -502,7 +506,12 @@
 c
 C    ---------------- Old parallel for minimization ----------------------
 c
-      if (imin.ne.0 .or. plevel.eq.0) then
+      if(imin.eq.2) then
+         idumm = 0
+      else
+         idumm = imin
+      endif
+      if (idumm.ne.0 .or. plevel.eq.0) then
```

```
         mpi_orig = .true.
         notdone = 1
       else
@@ -586,6 +595,17 @@
         endif
 c

+      elseif (IMIN.eq.2) then
+
+c     ** FILL THE GRID
+
+         call fillgrid(x,ix,ih,ipairs,X(LCRD),X(LWINV),X(Lmass)
+     $        ,X(LFORCE),X(Lvel),X(Lvel2),X(L45),X(Lcrdr),X(L50),X(L95
+     $        ),IX(I70),X(L75),erstop,r_stack,i_stack,X(LGST),X(LGFOR)
+     $        )
+
+         if (master) CALL MINRIT(X(LCRD),X(LGST),X(LGFOR),NGRID)
+
       ELSE
 C
 c (IMIN=1)
@@ -600,7 +620,7 @@
 c
 C           --- Write the restart file:
 c
-         if (master) CALL MINRIT(X(LCRD))
+         if (master) CALL MINRIT(X(LCRD),X(LGST),X(LGFOR),NGRID)
 C
       ENDIF
 c
--- sander_grid/set.f
+++ sander_grid/set.f
@@ -330,6 +330,7 @@
 #endif
 #include "extra.h"
 #include "nmr.h"
+#include "grid.h"
       integer target,fraction
       integer thismol, thismolsfirstnode, thismolslastnode
       integer thismolslastatom, group_world, group_temp
@@ -359,9 +360,11 @@
 c   --- do not stop before or after a single-atom resiude, since this might
 c       be a terminating hydrogen that would have a bond to another residue:
 c
-        if(ipres(ires).lt.gstart)then
+        if(ipres(ires).lt.gstart.AND.ipres(ires).ge.gstart+gx*gy*gz)then
             if ((ipres(ires+1) - ipres(ires)) .lt. 2) go to 10
```

```
             if ( ires.gt.1 .and.
      +          (ipres(ires) - ipres(ires-1)) .lt. 2) go to 10
+        endif
 c
 c   --- if this residue starts past the target atom, end the atom list
 c       at the end of the previous residue:
@@ -371,6 +374,8 @@
               go to 20
             end if
    10     continue
+          if(master) print*,node,nres,ires,ipres(ires),gstart,gstart+gx
+     $         *gy*gz
           write(6,*) 'Error in setpar: check code, input'
           call mexit(6,1)
    20   continue
--- sander_fluid/Makefile
+++ sander_fluid/Makefile
@@ -43,7 +43,7 @@
  ddotp.f random.f lsqfit.f amopen.f \
  debug.f ew_recip_reg.f ew_handle_dips.f ew_dipole_recip.f \
  mexit.f egb.f new_time.f extra_pts.f thermo_int.f \
-reorderwat.f matinv.f decomp.f
+ reorderwat.f matinv.f decomp.f fillgrid.f grow.f

 OBJ=  sander.o cshf.o noecalc.o noeread.o caldis.o \
       calrate.o dinten.o drates.o indexn.o kmat.o pearsn.o \
@@ -64,39 +64,14 @@
       ddotp.o random.o lsqfit.o amopen.o \
       debug.o ew_recip_reg.o ew_handle_dips.o ew_dipole_recip.o \
       mexit.o egb.o new_time.o extra_pts.o thermo_int.o \
-      reorderwat.o matinv.o decomp.o
+      reorderwat.o matinv.o decomp.o fillgrid.o grow.o

-LESOBJ=  sander.LES.o cshf.o noecalc.o noeread.o caldis.o \
-      calrate.o dinten.o drates.o indexn.o kmat.o pearsn.o \
-      plane.o remarc.o nmrcal.LES.o nmrred.LES.o \
-      restal.o getnat.o nmrnrg.o modwt.LES.o disnrg.o angnrg.o \
-      tornrg.o nmrprt.o nmrgrp.o nmrcms.o nmrcmf.o \
-      impnum.o nmrsht.o at2res.o chklin.o opnmrg.o \
-      printe.o runmin.o ndvprt.o force.LES.o rdparm.LES.o \
-      mdread.LES.o locmem.o runmd.LES.o newvel.o getcor.o \
-      r6ave.LES.o r6drv.o aveint.o degcnt.LES.o corf.o \
-      threeb.o tripl.o nmrrad.o decnvh.o \
-      fastwt.o echoin.o parallel.LES.o jnrg.LES.o \
-      shake.o ene.LES.o mdwrit.o minrit.o \
-      set.o setmm.o dynlib.LES.o mdfil.o nmlsrc.o \
```

```
-      ew_force.LES.o ew_setup.o ew_box.o ew_bspline.o\
-      ew_fft.o ew_direct.LES.o ew_recip.LES.o pcshift.o align.o \
-      rstack.o istack.o grdmax.o rfree.o rgroup.o \
-      ddotp.o random.o lsqfit.o amopen.o \
-      debug.o ew_recip_reg.o ew_handle_dips.o ew_dipole_recip.o \
-      mexit.o egb.o new_time.o extra_pts.LES.o \
-      thermo_int.o reorderwat.o matinv.o decomp.o
+all::    sander$(SFX)

-all::    sander$(SFX) sander.LES$(SFX)
-
 sander$(SFX): $(OBJ) syslib nxtsec lapack blas
  SYSLIB=`../sysdir lib` ; ../Compile LOAD -o sander$(SFX) \
  $(OBJ) ../lapack/lapack.a ../blas/blas.a ../lib/nxtsec.o $$SYSLIB;

-sander.LES$(SFX): $(LESOBJ) syslib nxtsec lapack blas
-SYSLIB=`../sysdir lib` ; ../Compile LOAD -o sander.LES$(SFX) \
-$(LESOBJ) ../lapack/lapack.a  ../blas/blas.a ../lib/nxtsec.o $$SYSLIB;
-
 bigsource:
  SYSDIR=`../sysdir dir`  ;  cd $$SYSDIR ; make sys.f
  SYSSRC=`../sysdir src` ;  ../Compile CPPONLY -DDPREC \
@@ -172,6 +147,8 @@
 egb.o: md.h
 egb.o: parms.h
 egb.o: def_time.h
+egb.o: grid.h
+egb.o: fluid.h
 egb.o: gbsa.h
 ene.o: dprec.h
 ene.LES.o: dprec.h
@@ -251,6 +228,10 @@
 ew_direct.LES.o: les.h
 ew_direct.o: ew_nbe.h
 ew_direct.LES.o: ew_nbe.h
+ew_direct.o: grid.h
+ew_direct.LES.o: grid.h
+ew_direct.o: fluid.h
+ew_direct.LES.o: fluid.h
 ew_direct.o: ew_directp.h
 ew_direct.LES.o: ew_directp.h
 ew_direct.o: ew_directe.h
@@ -374,6 +355,20 @@
 fastwt.o: vector.h
 fastwt.o: parallel.h
 fastwt.o: md.h
```

```
+fillgrid.o: dprec.h
+fillgrid.o: files.h
+fillgrid.o: md.h
+fillgrid.o: box.h
+fillgrid.o: nmr.h
+fillgrid.o: memory.h
+fillgrid.o: grid.h
+fillgrid.o: extra.h
+fillgrid.o: ew_cntrl.h
+fillgrid.o: ew_mpole.h
+fillgrid.o: def_time.h
+fillgrid.o: ew_unitcell.h
+fillgrid.o: extra_pts.h
+fillgrid.o: parallel.h
 force.o: dprec.h
 force.LES.o: dprec.h
 force.o: vector.h
@@ -412,13 +407,28 @@
 force.LES.o: extra.h
 force.o: tgtmd.h
 force.LES.o: tgtmd.h
+force.o: grid.h
+force.LES.o: grid.h
 force.o: les.h
 force.LES.o: les.h
 getcor.o: dprec.h
 getcor.o: files.h
 getcor.o: ew_cntrl.h
+getcor.o: grid.h
+getcor.o: parallel.h
+getcor.o: fluid.h
 getnat.o: dprec.h
 grdmax.o: dprec.h
+grow.o: dprec.h
+grow.o: vector.h
+grow.o: extra.h
+grow.o: grid.h
+grow.o: fluid.h
+grow.o: files.h
+grow.o: parallel.h
+grow.o: memory.h
+grow.o: md.h
+grow.o: parms.h
 impnum.o: dprec.h
 indexn.o: dprec.h
 indexn.o: nmr.h
```

```
@@ -479,6 +489,10 @@
 mdread.LES.o: sizes.h
 mdread.o: tgtmd.h
 mdread.LES.o: tgtmd.h
+mdread.o: grid.h
+mdread.LES.o: grid.h
+mdread.o: fluid.h
+mdread.LES.o: fluid.h
 mdread.o: les.h
 mdread.LES.o: les.h
 mdread.o: extra_pts.h
@@ -496,11 +510,15 @@
 mdwrit.o: dprec.h
 mdwrit.o: files.h
 mdwrit.o: ew_unitcell.h
+mdwrit.o: grid.h
+mdwrit.o: fluid.h
 minrit.o: dprec.h
 minrit.o: md.h
 minrit.o: files.h
 minrit.o: box.h
 minrit.o: ew_unitcell.h
+minrit.o: grid.h
+minrit.o: fluid.h
 modwt.o: dprec.h
 modwt.o: nmr.h
 ndvprt.o: dprec.h
@@ -604,6 +622,10 @@
 parallel.LES.o: new_time.h
 parallel.o: tgtmd.h
 parallel.LES.o: tgtmd.h
+parallel.o: grid.h
+parallel.LES.o: grid.h
+parallel.o: fluid.h
+parallel.LES.o: fluid.h
 pcshift.o: dprec.h
 pcshift.o: vector.h
 pcshift.o: nmr.h
@@ -653,6 +675,8 @@
 rdparm.LES.o: sizes.h
 rdparm.o: istack.h
 rdparm.LES.o: istack.h
+rdparm.o: grid.h
+rdparm.LES.o: grid.h
 remarc.o: dprec.h
 remarc.o: vector.h
```

```
 remarc.o: nmr.h
@@ -680,6 +704,8 @@
 runmd.o: def_time.h
 runmd.o: ew_unitcell.h
 runmd.o: extra_pts.h
+runmd.o: grid.h
+runmd.o: fluid.h
 runmd.o: les.h
 runmd.o: ew_localnb.h
 runmin.o: dprec.h
@@ -702,6 +728,8 @@
 sander.o: parms.h
 sander.o: extra.h
 sander.o: tgtmd.h
+sander.o: grid.h
+sander.o: fluid.h
 sander.o: les.h
 sander.o: parallel.h
 sander.o: ew_parallel.h
@@ -728,6 +756,7 @@
 set.o: parallel.h
 set.o: extra.h
 set.o: nmr.h
+set.o: grid.h
 setmm.o: dprec.h
 setmm.o: memory.h
 setmm.o: box.h
@@ -744,6 +773,9 @@
 tripl.o: dprec.h
 tripl.o: pol.h
 tripl.o: box.h
+yammpnb.o: dprec.h
+yammpnb.o: parallel.h
+yammpnb.o: md.h

 #-----------LIBS

@@ -922,10 +954,10 @@

 #-----------

-install: sander$(SFX) sander.LES$(SFX)
--/bin/mv sander$(SFX) sander.LES$(SFX) ../../exe
+install: sander$(SFX)
+ /bin/mv sander$(SFX) ../../exe/sander_fluid
```

```
 clean:
-../Clean sander$(SFX) sander.LES$(SFX)
+ ../Clean sander$(SFX)

 # DO NOT DELETE
--- sander_fluid/egb.f
+++ sander_fluid/egb.f
@@ -3,7 +3,7 @@
      $     epol,eelt,evdw,esurf,dvdl,vdwrad,ineighbor,p1,p2,p3,p4,
      $     r2x,rjx,vectmp1,vectmp2,vectmp3,vectmp4,vectmp5,
      $     vectmp6,vectmp7,vectmp8,jj,skipv,
-     $     k_vals,j_vals,psi,rbmax,rbmin,rbave,rbfluct)
+     $     k_vals,j_vals,psi,rbmax,rbmin,rbave,rbfluct,gstat)
 C
 C*************************************************************************
 C                                AMBER                                  **
@@ -104,6 +104,8 @@
 #include "md.h"
 #include "parms.h"
 #include "def_time.h"
+#include "grid.h"
+#include "fluid.h"
 c
       logical onstep,onstepi
       logical skipv(*)
@@ -128,6 +130,7 @@
      $       dedx,dedy,dedz,qi2h,temp7,daix,daiy,daiz,dij2i,datmp,dij3i,
      $       qid2h,si2,rj1i,dvdl,reff_i,psi,thi,thi2,
      $       dfac,beta,gamma,gamfac,fgbiorig
+      _REAL_ gstat(*)
 #ifdef ACE
       _REAL_ sigma,omega,vtilde,mu4,r3,r4
 #endif
@@ -199,6 +202,7 @@
 #else
       do i=1,natom
 #endif
+      if(gstat(i).ne.0.0) then
         xi = x(3*i-2)
         yi = x(3*i-1)
         zi = x(3*i)
@@ -217,6 +221,7 @@
         icount = 0
         do 25 j=i+1,natom
 c
+         if(gstat(j).ne.0.0.AND.i.ne.j) then
```

```
           xij = xi - x(3*j-2)
           yij = yi - x(3*j-1)
           zij = zi - x(3*j  )
@@ -226,6 +231,7 @@
           icount = icount + 1
           jj(icount) = j
           r2x(icount) = r2
+         endif
 c
    25   continue
 c
@@ -333,6 +339,7 @@
 c
         reff(i) = reff_i
 c
+        endif
       end do
 c
 #ifdef MPI
@@ -390,7 +397,6 @@
       call timer_stop(TIME_gbrad1)
 c
    90 continue
-c
 c-----------------------------------------------------------------------------
 c
 c     Step 2: loop over all pairs of atoms, computing the gas-phase
@@ -426,6 +432,7 @@
       iexcl = 1
       do i=1,maxi
 #endif
+      if(gstat(i).ne.0.0) then
         xi = x(3*i-2)
         yi = x(3*i-1)
         zi = x(3*i  )
@@ -450,6 +457,7 @@
         icount = 0
         do 150 j=i+1,natom
 c
+         if(gstat(j).ne.0.0.AND.i.ne.j) then
           xij = xi - x(3*j-2)
           yij = yi - x(3*j-1)
           zij = zi - x(3*j  )
@@ -461,6 +469,7 @@
           jj(icount) = j
           r2x(icount) = r2
```

```
           rjx(icount) = reff(j)
+         endif
 c
   150   continue
 #ifdef MASSLIB
@@ -567,7 +576,11 @@
           endif
           dl = intdieli - expmkf
 c
-          e = -qiqj*dl*fgbi
+          if(gstat(i).eq.2.0.OR.gstat(j).eq.2.0) then
+             e = 0.0
+          else
+             e = -qiqj*dl*fgbi
+          endif
           epol = epol + e
           if(idecomp.eq.1 .or. idecomp.eq.2) then
             call decpair(xx,ix,1,i,j,e)
@@ -644,6 +657,10 @@
               r6inv = r2inv*r2inv*r2inv
               f6 = cn2(ic)*r6inv
               f12 = cn1(ic)*(r6inv*r6inv)
+              if(gstat(i).eq.2.0.OR.gstat(j).eq.2.0) then
+                 f6 = f6*gam
+                 f12 = f12*gam
+              endif
               evdw = evdw + (f12 - f6)
               if(idecomp.eq.1 .or. idecomp.eq.2) then
                 call decpair(xx,ix,3,i,j,f12-f6)
@@ -700,6 +717,7 @@
 #else
         iexcl = iexcl + numex(i)
 #endif
+      endif
       end do
       call timer_stop(TIME_gbfrc)
 c
@@ -752,6 +770,7 @@
        do i=1,maxi
 #endif
 c
+      if(gstat(i).ne.1.0) then
           qi = charge(i)
           expmkf = exp( -kappa * reff(i) )*extdieli
           dl = intdieli - expmkf
@@ -785,6 +804,8 @@
```

```
           do 250 j=1,natom
             if( i.eq.j ) go to 250
 c
+           if(gstat(j).ne.0.0) then
+
              da(3*j-2) = 0.0d0
              da(3*j-1) = 0.0d0
              da(3*j  ) = 0.0d0
@@ -798,6 +819,7 @@
              icount = icount + 1
              jj(icount) = j
              r2x(icount) = r2
+           endif
 c
   250     continue
 #ifdef MASSLIB
@@ -927,6 +949,7 @@
             ineighbor(count)=0
           end if
 c
+          endif
         end do   ! end loop over atom i
 c
         if ( gbsa.eq.1 ) then
--- sander_fluid/ew_direct.f
+++ sander_fluid/ew_direct.f
@@ -693,7 +693,7 @@
      $     maxnblst,eelt,evdw,ehb,virial,eedvir,
      $     filter_cut,ee_type,eedmeth,dxdr,
      $     epol,dipole,field,mpoltype,r_stack,i_stack,
-     $     gindexlo,gindexhi)
+     $     gindexlo,gindexhi,gstat,ngrid)

       implicit none
 #ifdef MPI
@@ -723,6 +723,8 @@
       _REAL_ r_stack(*)

       integer nn
+      integer ngrid
+      _REAL_ gstat(*)

 c
 c     ---FLOW CONTROL FLAG (debug and future respa)
@@ -762,8 +764,7 @@
      $             eed_cub,eed_lin,charge,
```

```
      $             ntypes,iac,ico,cn1,cn2,asol,bsol,filter_cut,
      $             eelt,evdw,ehb,force,virial,ee_type,eedmeth,dxdr,
-     $             eedvir,r_stack(l_real),i_stack(l_int)
-     $                    )
+     $             eedvir,r_stack(l_real),i_stack(l_int),gstat)
               elseif ( mpoltype .eq. 1 )then
                call short_ene_dip(i,xk,yk,zk,ipairs(numpack),ntot,nvdw,
      $             bckptr,imagcrds,ewaldcof,eedtbdns,
@@ -1676,7 +1677,7 @@
      $        eed_cub,eed_lin,charge,
      $        ntypes,iac,ico,cn1,cn2,asol,bsol,filter_cut,
      $        eelt,evdw,ehb,frc,virial,
-     $        ee_type,eedmeth,dxdr,eedvir,tempre,tempint)
+     $        ee_type,eedmeth,dxdr,eedvir,tempre,tempint,gstat)
       implicit none
       _REAL_ xk,yk,zk,imagcrds(3,*)
       integer i,numvdw,numtot,bckptr(*)
@@ -1696,6 +1697,7 @@
       _REAL_ comm1
       _REAL_ xktran(3,18)
       _REAL_ e3dx,e4dx
+      _REAL_ gstat(*)
       integer mask27
       integer nprefetch
 #ifdef LES
@@ -1706,6 +1708,8 @@
 #include "parallel.h"
 #include "ew_tran.h"
 #include "ew_cntrl.h"
+#include "grid.h"
+#include "fluid.h"
       integer itran
 c
 c new variables.
--- sander_fluid/ew_directe.h
+++ sander_fluid/ew_directe.h
@@ -4,6 +4,7 @@
       do im_new = 1,icount
        j = tempint(im_new)

+        if(gstat(i).ne.0.0.AND.gstat(j).ne.0.0) then
         dfee = tempre(5*im_new-4)
         delx = tempre(5*im_new-3)
         dely = tempre(5*im_new-2)
@@ -20,12 +21,16 @@
           f6 = cn2(ic)*r6
```

```
           f12 = cn1(ic)*(r6*r6)
 #endif
-          evdw = evdw + f12 - f6
+          if(gstat(i).eq.2.0.OR.gstat(j).eq.2.0) then
+     f6 = f6*gam
+     f12 = f12*gam
+   endif
           df = dfee + (12.d0*f12 - 6.d0*f6)*delr2inv

           dfx = delx*df
           dfy = dely*df
           dfz = delz*df
+          evdw = evdw + f12 - f6
           vxx = vxx - dfx*delx
           vxy = vxy - dfx*dely
           vxz = vxz - dfx*delz
@@ -38,4 +43,5 @@
           frc(1,j) = frc(1,j) + dfx
           frc(2,j) = frc(2,j) + dfy
           frc(3,j) = frc(3,j) + dfz
+         endif
        enddo
--- sander_fluid/ew_directe2.h
+++ sander_fluid/ew_directe2.h
@@ -9,6 +9,7 @@
       do im_new = 1,icount
        j = tempint(im_new)

+        if(gstat(i).ne.0.0.AND.gstat(j).ne.0.0) then
         dfee = tempre(5*im_new-4)
         delx = tempre(5*im_new-3)
         dely = tempre(5*im_new-2)
@@ -26,7 +27,6 @@
           f10 = bsol(ic)*r10
           f12 = asol(ic)*(r10*delr2inv)
 #  endif
-          ehb = ehb + f12 - f10
           df = dfee + (12.d0*f12 - 10.d0*f10)*delr2inv
 #else
           df = dfee
@@ -34,6 +34,7 @@
           dfx = delx*df
           dfy = dely*df
           dfz = delz*df
+          ehb = ehb + f12 - f10
           vxx = vxx - dfx*delx
```

```
           vxy = vxy - dfx*dely
           vxz = vxz - dfx*delz
@@ -46,4 +47,5 @@
           frc(1,j) = frc(1,j) + dfx
           frc(2,j) = frc(2,j) + dfy
           frc(3,j) = frc(3,j) + dfz
+         endif
       end do
--- sander_fluid/ew_force.f
+++ sander_fluid/ew_force.f
@@ -294,7 +294,7 @@
      $       maxnblst,eed,evdw,ehb,dir_vir,eedvir,
      $       nbfilter,ee_type,eedmeth,dxdr,
      $       epold,X(linddip),X(lfield),mpoltype,r_stack,i_stack,
-     $       gindexlo,gindexhi)
+     $       gindexlo,gindexhi,X(lgst),ngrid)
 #ifdef MPI
         call MPI_COMM_RANK(MPI_COMM_WORLD,mytaskid,ierr)
         call MPI_COMM_SIZE(MPI_COMM_WORLD,numtasks,ierr)
--- sander_fluid/files.h
+++ sander_fluid/files.h
@@ -2,12 +2,12 @@
       character*80 mdin, mdout, inpcrd, parm, restrt,
      +             refc, mdvel, mden, mdcrd, mdinfo, nmr, mincor,
     +             vecs, radii, freqe,redir(8),
-     +             rstdip,mddip,inpdip
+     +             rstdip,mddip,inpdip,grid,grido,fluid
       character owrite
       common /files/ mdin, mdout, inpcrd, parm, restrt,
     +                refc, mdvel, mden, mdcrd, mdinfo, nmr, mincor,
     +                vecs, radii, freqe, owrite,
-     +             rstdip,mddip,inpdip
+     +             rstdip,mddip,inpdip,grid,grido,fluid

       integer       ITITL(20),ITITL1(20)
       COMMON/RUNHED/ITITL,    ITITL1
--- sander_fluid/fillgrid.f
+++ sander_fluid/fillgrid.f
@@ -0,0 +1,192 @@
+#include "dprec.h"
+      SUBROUTINE fillgrid(xx,ix,ih,ipairs,X,WINV,amass,F,V,VOLD,XR,XC
+     $     ,CONP,SKIP,NSP,TMA,erstop,r_stack,i_stack,gstat)
+
+      implicit none
+
+      integer   ipairs(*)
```

```
+      _REAL_ xx(*)
+      integer   ix(*), ih(*)
+      _REAL_ r_stack(*)
+      integer i_stack(*)
+#include "files.h"
+#include "md.h"
+#include "box.h"
+#include "nmr.h"
+#include "memory.h"
+#include "grid.h"
+#include "extra.h"
+#include "ew_cntrl.h"
+#include "ew_mpole.h"
+#include "def_time.h"
+#include "ew_unitcell.h"
+#include "extra_pts.h"
+#include "parallel.h"
+      logical do_list_update
+      LOGICAL SKIP(*),erstop
+      _REAL_ X(*),WINV(*),amass(*),F(*),V(*),VOLD(*),
+     $     XR(*),XC(*),CONP(*)
+      _REAL_ VIR(4),ENE(30)
+      _REAL_ TMA(*)
+      integer NSP(*)
+      _REAL_ gstat(*)
+
+      integer nix(6),niy(6),niz(6),test(ngrid)
+      integer i,j,k,counttry,countadd,point,pnx,pny,pnz,ixx,iyy,izz,io
+      real refene
+      character*80 sys
+      logical xp,xn,yp,yn,zp,zn
+
+      data nix/ -1, 1, 0, 0, 0, 0/
+      data niy/  0, 0,-1, 1, 0, 0/
+      data niz/  0, 0, 0, 0,-1, 1/
+
+c     ** Wipe test point array
+      do i = 1,ngrid
+         test(i) = 0
+      enddo
+      point = 0
+      xp = .false.
+      xn = .false.
+      yp = .false.
+      yn = .false.
+      zp = .false.
```

```
+      zn = .false.
+c     ** Add initial grid points to test array
+      do i = gstart,gstart+ngrid-1
+         if(gstat(i).eq.2.0) then
+            point = point + 1
+            test(point) = i
+            gstat(i) = 4.0
+         endif
+      enddo
+c     ** Calculate reference energy without any grid particles
+      CALL FORCE(xx,ix,ih,ipairs,X,F,ENE,VIR,r_stack,i_stack, xx(L96),
+     $     xx(L97), xx(L98), do_list_update)
+      refene = ene(1)
+      if(master) then
+         write(6,*)
+         write(6,'(a,f15.4)') 'Energy of system without grid',refene
+         call amflsh(6)
+      endif
+c     ** Loop over test point
+      counttry = 0
+      countadd = 0
+      do while(point.ne.0)
+         counttry = counttry + 1
+         i = test(point)
+         point = point-1
+c     ** Turn this grid particle on
+         gstat(i) = 1.0
+c     ** Calculate energy with this grid point turned on
+         CALL FORCE(xx,ix,ih,ipairs,X,F,ENE,VIR,r_stack,i_stack, xx(L96)
+     $        ,xx(L97), xx(L98), do_list_update)
+c     ** Determine grid location of this particle
+         j = i-gstart+1
+         pnx = int(real(j-1)/real(gy*gz))
+         pny = int(real(j-1-pnx*gy*gz)/real(gz))
+         pnz = j-pnx*gy*gz-pny*gz
+         pnx = pnx+1
+         pny = pny+1
+c     ** Write to output file
+         if(master) then
+            write(6,*)
+            write(6,'(a,i4,a,3i4)') 'Attempting to add grid point ',i
+     $           -gstart+1,' at ',pnx,pny,pnz
+            write(6,'(a,f15.4)') 'Energy with this point turned on '
+     $           ,ene(1)
+            write(6,'(a,f15.4)') 'Energy difference                '
+     $           ,ene(1)-refene
```

```
+         endif
+c     ** If energy increase is within acceptable levels
+         if(ene(1).le.refene+gtol) then
+            gstat(i) = 3.0
+            countadd = countadd + 1
+c     ** Loop over adjacent grid point
+            do k = 1,6
+               ixx = pnx+nix(k)
+               iyy = pny+niy(k)
+               izz = pnz+niz(k)
+c     ** Check for falling off the grid
+               if(ixx.eq.0) then
+                  xn = .TRUE.
+               elseif(ixx.gt.gx) then
+                  xp = .TRUE.
+               elseif(iyy.eq.0) then
+                  yn = .TRUE.
+               elseif(iyy.gt.gy) then
+                  yp = .TRUE.
+               elseif(izz.eq.0) then
+                  zn = .TRUE.
+               elseif(izz.gt.gz) then
+                  zp = .TRUE.
+               else
+c     ** Determine the grid point
+                  j = (ixx-1)*gy*gz + (iyy-1)*gz + izz + gstart-1
+c     ** If this grid point has not been considered previously
+                  if(gstat(j).eq.0.0) then
+c     ** Add to check list
+                     point = point + 1
+                     test(point) = j
+                     gstat(j) = 4.0
+                  endif
+               endif
+            enddo
+            if(master) write(6,'(a)') '    Added'
+         else
+            gstat(i) = 2.0
+            if(master) write(6,'(a)') 'Not Added'
+         endif
+         if(master) call amflsh(6)
+      enddo
+c     ** Write final summary
+      if(master) then
+         write(6,*)
+         write(6,'(a,i4,a,i4,a)') 'Added ',countadd
```

```
+     $        ,' grid points out of ',counttry,' attempted'
+      endif
+c     ** Turn all selected grid particles on
+      do i = gstart,gstart+ngrid-1
+         if(gstat(i).eq.3.0) gstat(i) = 1.0
+      enddo
+c     ** Calculate energy with selected grid particles turned on
+      CALL FORCE(xx,ix,ih,ipairs,X,F,ENE,VIR,r_stack,i_stack, xx(L96),
+     $     xx(L97), xx(L98), do_list_update)
+      if(master) then
+         write(6,*)
+         write(6,'(a,f15.4)') 'Energy of system with grid turned on'
+     $        ,ene(1)
+      endif
+
+c     ** WARNINGS FOR EDGE OF GRID BEING REACHED
+      if(master) then
+         if(xn) then
+            write(6,'(a)') 'Lower edge of grid reached in x direction'
+            print*,'Lower edge of grid reached in x direction'
+         endif
+         if(xp) then
+            write(6,'(a)') 'Upper edge of grid reached in x direction'
+            print*,'Upper edge of grid reached in x direction'
+         endif
+         if(yn) then
+            write(6,'(a)') 'Lower edge of grid reached in y direction'
+            print*,'Lower edge of grid reached in y direction'
+         endif
+         if(yp) then
+            write(6,'(a)') 'Upper edge of grid reached in y direction'
+            print*,'Upper edge of grid reached in y direction'
+         endif
+         if(zn) then
+            write(6,'(a)') 'Lower edge of grid reached in z direction'
+            print*,'Lower edge of grid reached in z direction'
+         endif
+         if(zp) then
+            write(6,'(a)') 'Upper edge of grid reached in z direction'
+            print*,'Upper edge of grid reached in z direction'
+         endif
+      endif
+
+
+
+      RETURN
```

```
+      END
--- sander_fluid/fluid.h
+++ sander_fluid/fluid.h
@@ -0,0 +1,3 @@
+        integer       ap,nta,rseed,nogrid,curgrow
+ real          gam,gammin,incgam
+ COMMON/FLUID/ap,nta,rseed,nogrid,curgrow,gam,gammin,incgam
--- sander_fluid/force.f
+++ sander_fluid/force.f
@@ -68,6 +68,7 @@
 #include "files.h"
 #include "extra.h"
 #include "tgtmd.h"
+#include "grid.h"
 #ifdef LES
 #include "les.h"
 #endif
@@ -87,11 +88,13 @@
      $       virvsene,eelt,e3bod,epol,esurf
       _REAL_ epolar,aveper,aveind,avetot,emtot,
      $       dipiter,dipole_temp
+      _REAL_ tforce(3*natom)
       integer l_r2x,l_rjx,l_tmp1,l_tmp2,l_tmp3,l_tmp4,l_tmp5,
      $        l_tmp6,l_tmp7,l_tmp8,l_jj,l_skipv
       integer l_kvls,l_jvls,l_psi
       integer l_da,l_sumd
       integer newbalance
+      integer j
       save newbalance
 C
       call timer_start(TIME_force)
@@ -418,7 +421,7 @@
      $      r_stack(l_tmp7),r_stack(l_tmp8),i_stack(l_jj),
      $      i_stack(l_skipv),
      $      i_stack(l_kvls),i_stack(l_jvls),r_stack(l_psi),
-     $      xx(L186),xx(L187),xx(L188),xx(L189) )
+     $      xx(L186),xx(L187),xx(L188),xx(L189),XX(lgst) )
         ene(2) = evdw
         ene(3) = eelt
         ene(4) = epol
--- sander_fluid/gbsa.h
+++ sander_fluid/gbsa.h
@@ -8,6 +8,7 @@
 #else
         do i=1,natom
 #endif
```

```
+   if(gstat(i).ne.0.0) then
           if (ineighbor(count).eq.0) then
             count=count+1
           else
@@ -199,6 +200,7 @@
             totsasa = totsasa + Ai
 c
           end if
+          endif
         end do
 c
 c       --- finished looping over i ---
--- sander_fluid/getcor.f
+++ sander_fluid/getcor.f
@@ -248,3 +248,29 @@
      +       /5x,'A     =',f10.5,'  B    =',f10.5,'  C     =',f10.5,/,
      +       /5x,'ALPHA =',f10.5,'  BETA =',f10.5,'  GAMMA =',f10.5,/)
       END
+
+
+      subroutine getgrid(natom,ngrid,gstat)
+      implicit none
+      integer i,natom,ngrid,stat(ngrid+1)
+      _REAL_ gstat(*)
+#include "grid.h"
+#include "parallel.h"
+#include "fluid.h"
+
+c     ** Read grid info
+      read(17,'(40i2)') (stat(i),i=1,ngrid)
+      read(17,'(e15.7)') gam
+
+c     ** All particles in the system interact....
+      do i = 1,natom
+         gstat(i) = 1.0
+      enddo
+c     ** Update grid particle info from the file......
+      do i = 1,ngrid
+         gstat(i+gstart-1) = real(stat(i))
+      enddo
+
+      return
+      end
--- sander_fluid/grid.h
+++ sander_fluid/grid.h
```

```
@@ -0,0 +1,3 @@
+        integer       gx,gy,gz,gstart
+ real          gdia,gtol
+ COMMON/BOX/gx,gy,gz,gstart,gdia,gtol
--- sander_fluid/grow.f
+++ sander_fluid/grow.f
@@ -0,0 +1,287 @@
+#include "dprec.h"
+#include "vector.h"
+c
+      SUBROUTINE init_grow(gstat,ngrid)
+
+      implicit none
+
+#include "extra.h"
+#include "grid.h"
+#include "fluid.h"
+#include "files.h"
+
+      _REAL_ gstat(*)
+      INTEGER ngrid,date_time(8),i
+      CHARACTER*12 real_clock(3)
+
+      if(master) then
+c     ** CREATE RANDOM NUMBER SEED FROM SYSTEM TIME
+         CALL DATE_AND_TIME(REAL_CLOCK(1),REAL_CLOCK(2),REAL_CLOCK(3)
+     $        ,DATE_TIME)
+         rseed = 0
+         do i = 1,8
+            rseed = rseed - date_time(i)
+         enddo
+c     ** Open output file
+         call amopen(20,fluid,owrite,'F','W')
+         write(20,3000)
+         write(20,3001)
+         write(20,3000)
+         call amflsh(20)
+c     ** Count number of fluid particles and find particle being grown
+         nogrid = 0
+         do i = gstart,gstart+ngrid
+            if(gstat(i).ne.0.0) then
+               nogrid = nogrid+1
+               if(gstat(i).eq.2.0) curgrow = i
+            endif
+         enddo
+      endif
```

```
+
+      return
+ 3000 format("------------------------------------------------------"
+     $     ,"------------")
+ 3001 format("- nstep - type - atom no. - nfluid   - Del EPtot    - "
+     $     ,"Gam/Dist   -")
+      end
+
+
+      SUBROUTINE grow(xx,ix,ih,ipairs,X,V,F,ENER,VIR,r_stack,i_stack,fs
+     $     ,rborn,reff,gstat,do_list_update,nstep,rndf,rndfp)
+
+      implicit none
+
+#ifdef MPI
+c     ========================== AMBER/MPI ===========================
+c
+c     NOTE: this routine contains MPI functionality to update the
+c     positions and velocities for all the atoms on a given node.
+c     After the positions are updated, communication (where necessary)
+c     is performed to update the coordinates and velocities on each
+c     processor.  Also note that wrappers to all the I/O routines are
+c     present to prevent all nodes except the master from executing
+c     I/O.  In addition, extra timing variables and calls are defined
+c     to allow profiling of this routine.
+c
+#  include "parallel.h"
+#ifdef MPI_DOUBLE_PRECISION
+#undef MPI_DOUBLE_PRECISION
+#endif
+#  include "mpif.h"
+#ifdef CRAY_PVP
+#define MPI_DOUBLE_PRECISION MPI_REAL8
+#endif
+      _REAL_ ekcmtt(3)
+#  ifdef T3D
+      common/ekcmtt/ekcmtt
+#  endif
+c     ======================== END AMBER/MPI =========================
+#endif
+#include "memory.h"
+#include "md.h"
+#include "parms.h"
+#include "extra.h"
+#include "grid.h"
+#include "fluid.h"
```

```
+
+      integer   ipairs(*)
+      _REAL_ xx(*)
+      integer   ix(*), ih(*)
+      _REAL_ fs(*),rborn(*),reff(*)
+      logical do_list_update,update
+      _REAL_ r_stack(*)
+      integer i_stack(*),nstep
+      _REAL_ X(3,*),V(3,*),F(*),ENE(30),ENER(*),VIR(*)
+      _REAL_ gstat(*)
+
+      integer i,j,count,aadd
+      real rndf,rndfp,refene,rsq_min,xi,yi,zi,rsq,c(3),r(2),r2,u(3),uran
+
+c     ** DETERMINE INITIAL ENERGY OF THE SYSTEM
+      CALL FORCE(xx,ix,ih,ipairs,X,F,ENE,VIR,
+     $     r_stack,i_stack,fs,rborn,reff,do_list_update)
+      refene = ene(1)
+
+c     ** CHECK THAT NO PARTICLES ARE TOO FAR AWAY FROM THE FLUID REGION
+      do i = gstart,gstart+ngrid-1
+         if(gstat(i).ne.0.0) then
+            rsq_min = 10000.0
+            do j = gstart,gstart+ngrid-1
+               if(gstat(j).ne.0.0.AND.i.ne.j) then
+                  xi = x(1,i)-x(1,j)
+                  yi = x(2,i)-x(2,j)
+                  zi = x(3,i)-x(3,j)
+                  rsq = xi*xi+yi*yi+zi*zi
+                  if(rsq.lt.rsq_min) rsq_min = rsq
+               endif
+            enddo
+c     ** If this particle is too far away then remove it
+            if(sqrt(rsq_min).gt.2.*gdia) then
+c     ** If its the particle being grown then we will start a new one
+               if(gstat(i).eq.2.0) gam = 1.5
+c     ** Zero velocity for the particle
+               v(1,i) = 0.0
+               v(2,i) = 0.0
+               v(3,i) = 0.0
+c     ** Change the number of degrees of freedom
+               rndf = rndf-3.d0
+               rndfp = rndfp-3.d0
+c     ** Turn off the particle
+               gstat(i) = 0.0
+c     ** Determine new energy
```

```
+               CALL FORCE(xx,ix,ih,ipairs,X,F,ENE,VIR,r_stack,i_stack,fs
+     $              ,rborn,reff,do_list_update)
+               nogrid = nogrid-1
+               if(master) then
+                  write(20,3000) nstep,i,nogrid,ene(1)-refene
+     $                 ,sqrt(rsq_min)
+                  call amflsh(20)
+               endif
+               refene = ene(1)
+            endif
+         endif
+      enddo
+
+      if(gam.lt.1.0) then
+c     ** We are  increasing the size of an existing particle
+         gam = gam*incgam
+         gam = min(gam,1.0)
+         CALL FORCE(xx,ix,ih,ipairs,X,F,ENE,VIR,r_stack,i_stack,fs,rborn
+     $        ,reff,do_list_update)
+         if(master) then
+            write(20,3007) nstep,curgrow,nogrid,refene-ene(1),gam
+            call amflsh(20)
+         endif
+      else
+c     ** We are adding a new particle.....
+         gam = gammin
+         rndf = rndf+3.d0
+         rndfp = rndfp+3.d0
+         aadd = 0
+c     ** Set particle currently being grown to stat 1
+         do i = gstart,gstart+ngrid-1
+            if(gstat(i).eq.2.0) gstat(i) = 1.0
+         enddo
+c     ** Find particle to add
+         j = 0
+         i = gstart
+         do while(j.eq.0)
+            if(gstat(i).eq.0.0) j = i
+            i = i+1
+            if(i.eq.gstart+ngrid) then
+               if(master) write(20,*)
+               if(master) write(20,3001)
+               if(master) write(6,*)
+               if(master) write(6,3001)
+               call mexit(6,1)
+            endif
```

```
+         enddo
+         gstat(j) = 2.0
+c     ** DETERMINE COM OF THE FLUID
+         c(1) = 0.0
+         c(2) = 0.0
+         c(3) = 0.0
+         count = 0
+         do i = gstart,gstart+ngrid-1
+            if(gstat(i).ne.0.0) then
+               count = count+1
+               c(1) = c(1) + x(1,i)
+               c(2) = c(2) + x(2,i)
+               c(3) = c(3) + x(3,i)
+            endif
+         enddo
+         c(1) = c(1)/real(count)
+         c(2) = c(2)/real(count)
+         c(3) = c(3)/real(count)
+c     ** GENERATE A UNIT VECTOR IN A RANDOM DIRECTION
+ 2001    r(1) = 1.-2.*uran(rseed)
+         r(2) = 1.-2.*uran(rseed)
+         r2 = r(1)*r(1)+r(2)*r(2)
+         if(r2.lt.1.0) then
+            u(1) = 2.*r(1)*sqrt(1.-r2)
+            u(2) = 2.*r(2)*sqrt(1.-r2)
+            u(3) = 1.-2.*r2
+         else
+            goto 2001
+         endif
+c     ** THIS VECTOR IS THEN USED TO GENERATE A POINT IN A SPHERE
+c     ** CENTERED ON CX WITH RADIUS 0.5*gdia....
+         u(1) = u(1)*0.5*gdia
+         u(2) = u(2)*0.5*gdia
+         u(3) = u(3)*0.5*gdia
+c     ** ....AND THE NEW PARTICLE IS LOCATED THERE
+         x(1,j) = c(1)+u(1)
+         x(2,j) = c(2)+u(2)
+         x(3,j) = c(3)+u(3)
+c     ** CALCULATE ENERGY IN THIS LOCATION
+         CALL FORCE(xx,ix,ih,ipairs,X,F,ENE,VIR,r_stack,i_stack,fs,rborn
+     $        ,reff,do_list_update)
+c     ** IF THE ENERGY CHANGE IS HUGELY POSITIVE THEN TRY AGAIN
+         if(ene(1)-refene.gt.gtol) then
+            if(master) then
+               write(20,3008) nstep,j,nogrid,ene(1)-refene,gam
+               call amflsh(20)
```

```
+            endif
+            aadd = aadd+1
+            if(aadd.lt.100) then
+               goto 2001
+            else
+               if(master) write(20,3002)
+               if(master) write(6,3002)
+               call mexit(6,1)
+            endif
+         elseif(master) then
+            nogrid = nogrid+1
+            write(20,3006) nstep,j,nogrid,ene(1)-refene,gam
+            call amflsh(20)
+         endif
+         curgrow = j
+      endif
+
+
+      return
+ 3000 format(i7,'   REM  ',2x,2(i7,3x),f12.4,3x,f12.4)
+ 3001 format("NO UNASIGNED PARTICLES TO ADD - EXITING")
+ 3002 format("ATTEMPTED TO ADD PARTICLE 100 TIMES - EXITING")
+ 3006 format(i7,'   ADD  ',2x,2(i7,3x),f12.4,3x,e12.5)
+ 3007 format(i7,'   GROW ',2x,2(i7,3x),f12.4,3x,e12.5)
+ 3008 format(i7,'   NADD ',2x,2(i7,3x),f12.4,3x,e12.5)
+      end
+
+
+
+
+      FUNCTION uran(idum)
+      INTEGER idum,IM1,IM2,IMM1,IA1,IA2,IQ1,IQ2,IR1,IR2,NTAB,NDIV
+      REAL uran,AM,EPS,RNMX
+      PARAMETER (IM1=2147483563,IM2=2147483399,AM=1./IM1,IMM1=IM1-1,
+     *IA1=40014,IA2=40692,IQ1=53668,IQ2=52774,IR1=12211,IR2=3791,
+     *NTAB=32,NDIV=1+IMM1/NTAB,EPS=1.2e-7,RNMX=1.-EPS)
+      INTEGER idum2,j,k,iv(NTAB),iy
+      SAVE iv,iy,idum2
+      DATA idum2/123456789/, iv/NTAB*0/, iy/0/
+      if (idum.le.0) then
+        idum=max(-idum,1)
+        idum2=idum
+        do 11 j=NTAB+8,1,-1
+          k=idum/IQ1
+          idum=IA1*(idum-k*IQ1)-k*IR1
+          if (idum.lt.0) idum=idum+IM1
```

```
+          if (j.le.NTAB) iv(j)=idum
+11      continue
+        iy=iv(1)
+      endif
+      k=idum/IQ1
+      idum=IA1*(idum-k*IQ1)-k*IR1
+      if (idum.lt.0) idum=idum+IM1
+      k=idum2/IQ2
+      idum2=IA2*(idum2-k*IQ2)-k*IR2
+      if (idum2.lt.0) idum2=idum2+IM2
+      j=1+iy/NDIV
+      iy=iv(j)-idum2
+      iv(j)=idum
+      if(iy.lt.1)iy=iy+IMM1
+      uran=min(AM*iy,RNMX)
+      return
+      END
+C  (C) Copr. 1986-92 Numerical Recipes Software ]uk<3jn>#S.sXz%.
--- sander_fluid/locmem.f
+++ sander_fluid/locmem.f
@@ -52,6 +52,7 @@
 C     AMASS   ...  LWINV ! ATOMIC MASSES (inverted in rdparm - see Lmass)
 C     XCHRG   ...  Lpol  ! atomic polarizibilities
 C     C       ...  LCRD  ! COORDINATES
+c     c            Lgst  ! Status of grid particle
 C     F       ...  Lforce! FORCE
 C     V       ...  Lvel  ! VELOCITY for MD, work space for min
 C     VOLD    ...  Lvel2 ! OLD VELOCITY for MD
@@ -201,6 +202,7 @@
         end if
       endif
       call adj_mem_ptr( r_ptr, LCRD, 3*natom + MXVAR )
+      call adj_mem_ptr( r_ptr, LGST, natom )
       call adj_mem_ptr( r_ptr, Lforce, 3*natom + MXVAR + 40 )
       if (imin.eq.0) then
         call adj_mem_ptr( r_ptr, Lvel,  3*natom + MXVAR )
@@ -476,10 +478,11 @@
         MAXPR = 1
       else
         if( numextra.eq. 0 ) then
-          maxpr_float = natom * (cutoffnb + skinnb)**3 / 3.5d0
+          maxpr_float = 10.0*natom * (cutoffnb + skinnb)**3 / 3.5d0
         else   ! need more nonbon storage with extra points
-          maxpr_float = natom * (cutoffnb + skinnb)**3 / 2.5d0
+          maxpr_float = 10.0*natom * (cutoffnb + skinnb)**3 / 2.5d0
         end if
```

```
+
 c
 c       --- cap at maximum possible number of pairs:
 c
--- sander_fluid/mdfil.f
+++ sander_fluid/mdfil.f
@@ -55,6 +55,9 @@
       inpdip = 'inpdip'
       mddip = 'mddip'
       radii = 'radii'
+      grid = 'grid'
+      grido = 'grido'
+      fluid = 'fluid'
 c
 c     --- default status of output: new
 c
@@ -142,6 +145,15 @@
             elseif (arg .eq. '-mdip') then
                  iarg = iarg + 1
                  call getarg(iarg,mddip)
+            elseif (arg .eq. '-gi') then
+                 iarg = iarg + 1
+                 call getarg(iarg,grid)
+            elseif (arg .eq. '-go') then
+                 iarg = iarg + 1
+                 call getarg(iarg,grido)
+            elseif (arg .eq. '-fl') then
+                 iarg = iarg + 1
+                 call getarg(iarg,fluid)
 #ifdef MPI
             elseif (arg .eq. '-p4pg') then
                  iarg = iarg+1
@@ -201,5 +213,5 @@
      +   'usage: sander  [-O] -i mdin -o mdout -p prmtop -c inpcrd ',
      +   '-r restrt',/19x,'[-ref refc -x mdcrd -v mdvel -e mden ',
      +   '-idip inpdip -rdip rstdip -mdip mddip ',
-     +   '-inf mdinfo -radii radii]')
+     +   '-inf mdinfo -radii radii -gi grid_in -go grid_out -fl fluid]')
       end
--- sander_fluid/mdread.f
+++ sander_fluid/mdread.f
@@ -33,6 +33,8 @@
 #include "def_time.h"
 #include "sizes.h"
 #include "tgtmd.h"
+#include "grid.h"
```

```
+#include "fluid.h"
 #ifdef LES
 #include "les.h"
 #endif
@@ -68,7 +70,7 @@
      *          surften,iwrap,nrespa,nrespai,gamma_ln,extdiel,intdiel,
      *          cut_inner,icfe,clambda,klambda,
      *          rbornstat,lastrst,lastist,itgtmd,tgtrmsd,tgtmdfrc,
-     *          idecomp,temp0les
+     *          idecomp,temp0les,gtol,ap,gammin,incgam,nta
 C
 C Define default water residue name and the names of water oxygen & hydrogens
 C
@@ -90,7 +92,8 @@
      *              'RESTRT',RESTRT(1:70) , 'REFC'  ,REFC(1:70)  ,
      *              'MDVEL' ,MDVEL(1:70)  , 'MDEN'   ,MDEN(1:70) ,
      *              'MDCRD' ,MDCRD(1:70)  , 'MDINFO' ,MDINFO(1:70),
-     *              'INPDIP', INPDIP(1:70), 'RSTDIP', RSTDIP(1:70)
+     *              'INPDIP', INPDIP(1:70), 'RSTDIP', RSTDIP(1:70),
+     *              'GRID', GRID(1:70), 'GRIDO', GRIDO(1:70)
 C
 C Echo the input file to the user:
 C
@@ -218,6 +221,17 @@
       idecomp = 0
       lastrst = MAX_RSTACK
       lastist = MAX_ISTACK
+c
+c     tollerance for adding grid particles
+c
+      gtol = 0.0
+c
+c     particle growing stuff
+c
+      ap = 0
+      nta = 100
+      gammin = 1.90735d-6
+      incgam = 2.0
 C
 C     Check to see if "cntrl" namelist has been defined.
 C
@@ -367,7 +381,7 @@
      +       'Amber 7  SANDER                    Scripps/UCSF 2002',
      +       /10X,55(1H-)/)
  9309 FORMAT(/80(1H-)/'   1.  RESOURCE   USE: ',/80(1H-)/)
- 9700 FORMAT(/,'File Assignments:',/,12('|',A6,': ',A,/))
```

```
+ 9700 FORMAT(/,'File Assignments:',/,14('|',A6,': ',A,/))
       end
 c
 c
@@ -414,6 +428,8 @@
 #include "ew_legal.h"
 #include "def_time.h"
 #include "tgtmd.h"
+#include "grid.h"
+#include "fluid.h"
 #ifdef LES
 #include "les.h"
 #endif
@@ -474,7 +490,8 @@
       WRITE(6,9328)
       WRITE(6,9008) ITITL
       write(6,'(/a)') 'General flags:'
-      write(6,'(5x,2(a,i8))') 'imin    =',imin,', nmropt  =',nmropt
+      write(6,'(5x,3(a,i8))') 'imin    =',imin,', nmropt  =',nmropt
+     .     ,', ap      =',ap

       write(6,'(/a)') 'Nature and format of input:'
       write(6,'(5x,4(a,i8))') 'ntx     =',ntx,', irest   =',irest,
@@ -505,6 +522,19 @@
       write(6,'(5x,3(a,f10.5))') 'scnb    =',scnb,
      .     ', scee    =',scee
 c
+c
+      if( imin.eq.2 ) then
+        write(6,'(/a)') 'Energy tolerance for inserted particles:'
+        write(6,'(5x,1(a,f10.5))') 'gtol    =',gtol
+      endif
+      if( ap.eq.1 ) then
+        write(6,'(/a)') 'Particle grow parameters:'
+        write(6,'(5x,1(a,i8),2(a,f10.5))') 'nta     =',nta,', gtol    ='
+     $       ,gtol
+        write(6,'(5x,2(a,e12.5))') 'gammin  =',gammin,', incgam  ='
+     $       ,incgam
+      endif
+c
       write(6,'(/a)') 'Frozen or restrained atoms:'
       write(6,'(5x,4(a,i8))') 'ibelly  =',ibelly,', ntr     =',ntr

@@ -839,6 +869,20 @@
               x(L175-1+i) = -0.16038d0
               x(L180-1+i) = -0.00015512d0
```

```
              x(L185-1+i) = 0.00016453d0
+            else if (atype.eq.'FE') then
+c ** MADE UP BY IMW 11/6/04
+               x(L165-1+i) = 1.80d0 + 1.4d0
+               x(L170-1+i) = 0.070344d0
+               x(L175-1+i) = -0.019015d0
+               x(L180-1+i) = -0.000022009d0
+               x(L185-1+i) = 0.000016875d0
+            else if (atype.eq.'LJ') then
+c ** MADE UP BY IMW 11/6/04
+               x(L165-1+i) = 0.25*gdia + 1.4d0
+               x(L170-1+i) = 1.0d0
+               x(L175-1+i) = 0.0d0
+               x(L180-1+i) = 0.0d0
+               x(L185-1+i) = 0.0d0
             else
               write( 0,* ) 'bad atom type: ',atype
               call mexit( 6,1 )
--- sander_fluid/mdwrit.f
+++ sander_fluid/mdwrit.f
@@ -1,6 +1,6 @@
 #include "dprec.h"
       SUBROUTINE MDWRIT(nstep,NRP,NR,NRES,NTXO,NTR,NTB,X,V,XC,
-     +     BOX,IGRAPH,LBRES,IPRES,TT)
+     +     BOX,IGRAPH,LBRES,IPRES,TT,ngrid,gstat)
 C
 C*********************************************************************
 C                               AMBER                                **
@@ -21,11 +21,11 @@
 C*********************************************************************
 C
       implicit none
-      integer nstep,NRP,NR,NRES,NTXO,NTR,NTB,IGRAPH,LBRES,IPRES
-      _REAL_ X,V,XC,BOX,TT
+      integer nstep,NRP,NR,NRES,NTXO,NTR,NTB,IGRAPH,LBRES,IPRES,ngrid
+      _REAL_ X,V,XC,BOX,TT,gstat
       character*89 restrt2
       character*12 num
-      integer istart,iend
+      integer istart,iend,i,j
       logical first
       save first
       data first/.true./
@@ -36,20 +36,27 @@
       if( first ) then
         if (ntxo.eq.0) then
```

```
           call amopen(16,restrt,owrite,'U','W')
+          call amopen(18,grido,owrite,'U','W')
         else
           call amopen(16,restrt,owrite,'F','W')
+          call amopen(18,grido,owrite,'F','W')
         endif
         first = .false.
       else
         if (ntxo.eq.0) then
           call amopen(16,restrt,'O','U','W')
+          call amopen(18,grido,'O','U','W')
         else
           call amopen(16,restrt,'O','F','W')
+          call amopen(18,grido,'O','F','W')
         endif
       endif
       CALL MDWRI2(16,NRP,NR,NRES,NTXO,NTR,NTB,X,V,XC,
      +                  BOX,IGRAPH,LBRES,IPRES,TT)
+      CALL MDWRI3(ngrid,gstat)
+
       close(16)
+      close(18)
 c
 c     -- consider whether to save secondary restrt;
 c
@@ -124,3 +131,23 @@
  9028 FORMAT(6F12.7)
       RETURN
       END
+
+
+      subroutine mdwri3(ngrid,gstat)
+      implicit none
+#include "grid.h"
+#include "fluid.h"
+      integer i,ngrid,stat(ngrid)
+      _REAL_ gstat(*)
+
+c     ** WRITE OUT GRID STUFF
+      do i = 1,ngrid
+         stat(i)= int(gstat(i+gstart-1))
+      enddo
+
+      write(18,'(4i6,f8.3)') ngrid,gx,gy,gz,gdia
+      write(18,'(40i2)')    (stat(i),i=1,ngrid)
+      write(18,'(e15.7)') gam
```

```
+
+      return
+      end
--- sander_fluid/memory.h
+++ sander_fluid/memory.h
@@ -3,7 +3,7 @@
 c      more variables to the "locmem" style of memory management
 c
 c       BC_MEMORY is the size of the MEMORY common block:
-#define BC_MEMORY 171
+#define BC_MEMORY 173
 c
       integer       NATOM,NRES,NBONH,NBONA,NTHETH,NTHETA,NPHIH,
      +    NPHIA,NNB,NTYPES,NRCP,NCONP,MAXMEM,NWDVAR,MAXNB,NPARM,
@@ -23,7 +23,8 @@
      +    L240,L241,L242,
      +    m02,m04,m06,m08,m10,m12,m14,m16,m18,i01,
      +    IVM01,IVM02,nrealb,nintb,nholb,npairb,lastr,lasti,lasth,
-     +    lastpr,lastrst,lastist,nbper,ngper,ndper,ifpert,Lpolp
+     +    lastpr,lastrst,lastist,nbper,ngper,ndper,ifpert,Lpolp,
+     +    Ngrid,LGST

       COMMON/MEMORY/NATOM,NRES,NBONH,NBONA,NTHETH,NTHETA,NPHIH,
      +    NPHIA,NNB,NTYPES,NRCP,NCONP,MAXMEM,NWDVAR,MAXNB,NPARM,
@@ -43,4 +44,5 @@
      +    L240,L241,L242,
      +    m02,m04,m06,m08,m10,m12,m14,m16,m18,i01,
      +    IVM01,IVM02,nrealb,nintb,nholb,npairb,lastr,lasti,lasth,
-     +    lastpr,lastrst,lastist,nbper,ngper,ndper,ifpert,Lpolp
+     +    lastpr,lastrst,lastist,nbper,ngper,ndper,ifpert,Lpolp,
+     +    Ngrid,LGST
--- sander_fluid/minrit.f
+++ sander_fluid/minrit.f
@@ -1,5 +1,5 @@
 #include "dprec.h"
-      SUBROUTINE MINRIT(X)
+      SUBROUTINE MINRIT(X,gstat,ngrid)
 C
 C*****************************************************************************
 C                               AMBER                                       **
@@ -27,7 +27,10 @@
 #include "files.h"
 #include "box.h"
 #include "ew_unitcell.h"
-      DIMENSION X(*)
+#include "grid.h"
```

```
+#include "fluid.h"
+      DIMENSION X(*),gstat(*)
+      integer stat(ngrid)
 C
       NR = NRP
       NR3 = 3*NR
@@ -34,8 +37,10 @@

       if (ntxo.le.0) then
            call amopen(16,restrt,owrite,'U','W')
+           call amopen(18,grido,owrite,'U','W')
       else
            call amopen(16,restrt,owrite,'F','W')
+           call amopen(18,grido,owrite,'F','W')
       endif
 C
       IF (NTXO.ne.0) THEN
@@ -53,7 +58,15 @@
           WRITE(16) NR,DUMM
           WRITE(16) (X(I),I = 1,NR3)
       ENDIF
+c     ** WRITE OUT GRID STUFF
+      do i = 1,ngrid
+         stat(i)= int(gstat(i+gstart-1))
+      enddo
+      write(18,'(4i6,f8.3)') ngrid,gx,gy,gz,gdia
+      write(18,'(40i2)')    (stat(i),i=1,ngrid)
+      write(18,'(e15.7)') gam
       close(unit=16)
+      close(unit=18)
    40 FORMAT(20A4)
   220 FORMAT(I5,F10.5)
   221 FORMAT(I6,F10.5)
--- sander_fluid/parallel.f
+++ sander_fluid/parallel.f
@@ -66,11 +66,25 @@
 #include "istack.h"
 #include "new_time.h"
 #include "tgtmd.h"
+#include "grid.h"
+#include "fluid.h"
 c
       dimension xx(*),ix(*),ih(*)
 c
 c     Send and receive common blocks from the master node:
 c
```

```
+c  fluid.h
+c
+      call mpi_bcast(ap,5,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
+      call mpi_bcast(gam,3,MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ierr)
+      call MPI_BARRIER(MPI_COMM_WORLD,ierr)
+c
+c  grid.h
+c
+      call mpi_bcast(gx,4,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
+      call mpi_bcast(gdia,2,MPI_DOUBLE_PRECISION,0,MPI_COMM_WORLD,ierr)
+      call MPI_BARRIER(MPI_COMM_WORLD,ierr)
+c
 c  files.h:
 c
       call mpi_bcast(NTPR,BC_HULP,MPI_INTEGER,0,MPI_COMM_WORLD,ierr)
@@ -268,7 +282,8 @@
       return
       end
 c-----------------------------------------------------------------------
-      subroutine fdist(f,forcetmp,ene,vir,npair,nhb,r_stack,newbalance)
+      subroutine fdist(f,forcetmp,ene,vir,npair,nhb
+     $     ,r_stack,newbalance)
 c*************************************************************
 c   for the "original version", (when mpi_orig is set):
 c
@@ -346,7 +361,7 @@
           f(i) = forcetmp(i)
         enddo
       else
-c
+cc
 c       ---Add all copies of virial and energy and put result back on ALL nodes:
 c
         call mpi_allreduce(f(j),forcetmp(j),35,
--- sander_fluid/rdparm.f
+++ sander_fluid/rdparm.f
@@ -123,6 +123,7 @@
 #include "pol.h"
 #include "sizes.h"
 #include "istack.h"
+#include "grid.h"
 #ifdef LES
 #  include "les.h"
       integer iexcl,numex,k1,j1
@@ -756,6 +757,20 @@
       read(nf,9128) (rub(i),i=1,numang)
```

```
 #endif
 C
+c     ** Find the first grid particle
+      gstart = 0
+      i = 1
+      do while(gstart.eq.0)
+         if(iH(m04+I-1).EQ."LJ  ") gstart = i
+         i = i+1
+      enddo
+c     ** Overwrite born radius of LJ particles with real radius
+      if(igb.ne.0) then
+         do i = gstart,natom
+           x(L97+i-1) = 0.5*gdia
+        enddo
+      endif
+
       RETURN
  9108 FORMAT(20A4)
  9128 FORMAT(5E16.8)
--- sander_fluid/runmd.f
+++ sander_fluid/runmd.f
@@ -64,6 +64,8 @@
 #include "def_time.h"
 #include "ew_unitcell.h"
 #include "extra_pts.h"
+#include "grid.h"
+#include "fluid.h"
 #ifdef LES
 #  include "les.h"
 #endif
@@ -82,7 +84,7 @@
 C
 C
       logical do_list_update
-      LOGICAL SKIP(*),BELLY,LOUT,LOUTFM,erstop,vlim,onstep
+      LOGICAL SKIP(*),BELLY,LOUT,LOUTFM,erstop,vlim,onstep,lgrow
       _REAL_ X(*),WINV(*),amass(*),F(*),V(*),VOLD(*),
      $     XR(*),XC(*),CONP(*),VOL
       _REAL_ ENERT(51),ENERT2(51),ENER(51),VIR(4),EKCMT(4)
@@ -89,7 +91,7 @@
       _REAL_ enert_old(51),enert2_old(51),ecopy(51),edvdl(51),
      $      enert_tmp(51),enert2_tmp(51),etot_start,ezb,edvdl_r(51)
       _REAL_ PRES(4),RMU(3),FAC(3),vircopy(3),clfac
-      _REAL_ TMA(*)
+      _REAL_ TMA(*),Y(natom*3)
```

```
       _REAL_ tspan,atempdrop,fln,scaltp
       _REAL_ vel,vel2,vcmx,vcmy,vcmz,vmax,vx,vy,vz
@@ -118,6 +120,7 @@
       _REAL_ xcen,ycen,zcen,extents(3,2),centertest
 C
       _REAL_ small
+      _REAL_ refx,refy,refz
       DATA SMALL/1.0D-7/
       DATA NREN/51/
 c
@@ -235,6 +238,12 @@
 C RNDFS = # degrees of freedom for solvent
 C RNDF = total number of degrees of freedom.
 C
+c     Reduce RNDFP to account for particles not present
+      do i = 1,ngrid
+         if(XX(lgst+i+gstart-1).eq.0.0) then
+            rndfp = rndfp-3.
+         endif
+      enddo
 C    modify RNDFP to reflect NDFMIN (set in mdread)
 C
       RNDFP = RNDFP - NDFMIN
@@ -1428,6 +1437,20 @@
       if (mod(nstep,nsnb) .eq. 0) ntnb = 1
       lout = mod(nstep,ntpr) .eq. 0 .and. onstep
       irespa = irespa + 1
+c---------------------------------------------------------------------
+c  Step 8.5: add or grow particle
+c---------------------------------------------------------------------
+      lgrow = mod(nstep,nta) .eq. 0 .and. onstep
+      if(LGROW.AND.ap.ne.0) then
+         call grow(xx,ix,ih,ipairs,X,V,F,ENER(23),VIR,r_stack,i_stack
+     $        ,xx(L96),xx(L97),xx(L98),XX(lgst),do_list_update,nstep
+     $        ,rndf,rndfp)
+         FAC(1) = BOLTZ2*RNDF
+         FAC(2) = BOLTZ2*RNDFP
+         ekin0  = fac(1)*temp0
+         EKINP0 = FAC(2)*TEMP0
+         DTMP0  = FAC(1)*DTEMP
+      endif
 C
 c---------------------------------------------------------------------
 C  Step 9:  output from this step if required:
@@ -1463,12 +1486,14 @@
               if(ireorderwat.eq.1)then
```

```
                  CALL MDWRIT(nstep,NRP,NR,NRES,NTXO,NTR,NTB,
      $                r_stack(ltmpX),r_stack(ltmpv),
-     $                XX(Lcrdr),BOX,ih(m04),ih(m02),IX(IO2),T)
+     $                XX(Lcrdr),BOX,ih(m04),ih(m02),IX(IO2),T,
+     $                ngrid,XX(Lgst))
                  else
 #endif
                  CALL MDWRIT(nstep,NRP,NR,NRES,NTXO,NTR,NTB,
      $             X,V,
-     $             XX(Lcrdr),BOX,ih(m04),ih(m02),IX(IO2),T)
+     $             XX(Lcrdr),BOX,ih(m04),ih(m02),IX(IO2),T,
+     $             ngrid,XX(Lgst))
 #ifdef ROWAT
                  endif
 #endif
@@ -1500,12 +1525,14 @@
                  if(ireorderwat.eq.1)then
                     CALL MDWRIT(nstep,NRP,NR,NRES,NTXO,NTR,NTB,
      +                r_stack(l_temp),r_stack(ltmpV),
-     $                XX(Lcrdr),BOX,ih(m04),ih(m02),IX(IO2),T)
+     $                XX(Lcrdr),BOX,ih(m04),ih(m02),IX(IO2),T,
+     $                ngrid,XX(Lgst))
                  else
 #endif
                     CALL MDWRIT(nstep,NRP,NR,NRES,NTXO,NTR,NTB,
      +                r_stack(l_temp),  V,
-     $                XX(Lcrdr),BOX,ih(m04),ih(m02),IX(IO2),T)
+     $                XX(Lcrdr),BOX,ih(m04),ih(m02),IX(IO2),T,
+     $                ngrid,XX(Lgst))
 #ifdef ROWAT
                  endif
 #endif
@@ -1528,6 +1555,24 @@
 C     -- Coordinate archive:
 C
          IF (itdump) then
+c     ** Find an active grid particle
+            do i = gstart,gstart+ngrid-1
+               if(XX(Lgst+i-1).eq.1.0) j = i
+            enddo
+            refx = x((j-1)*3+1)
+            refy = x((j-1)*3+2)
+            refz = x((j-1)*3+3)
+            do i = 1,natom
+               if(XX(Lgst+i-1).eq.0.0) then
+                  y((i-1)*3+1) = refx
```

```
+                  y((i-1)*3+2) = refy
+                  y((i-1)*3+3) = refz
+               else
+                  y((i-1)*3+1) = x((i-1)*3+1)
+                  y((i-1)*3+2) = x((i-1)*3+2)
+                  y((i-1)*3+3) = x((i-1)*3+3)
+               endif
+            enddo
 #ifdef ROWAT
             if(.not.tmpfilled .and. ireorderwat.eq.1)then
                call get_stack(ltmpx,nr3)
@@ -1545,7 +1590,7 @@
                   CALL CORPAC(r_stack(ltmpX),1,NRX,12,LOUTFM)
                else
 #endif
-                  CALL CORPAC(X,1,NRX,12,LOUTFM)
+                  CALL CORPAC(Y,1,NRX,12,LOUTFM)
 #ifdef ROWAT
                endif
 #endif
--- sander_fluid/sander.f
+++ sander_fluid/sander.f
@@ -39,6 +39,8 @@
 #include "parms.h"
 #include "extra.h"
 #include "tgtmd.h"
+#include "grid.h"
+#include "fluid.h"
 #ifdef LES
 #include "les.h"
 #endif
@@ -187,6 +189,8 @@
          CALL MDREAD1()
          call amopen(8,parm,'O','F','R')
          CALL RDPARM1(8)
+         call amopen(17,grid,'O','F','R')
+         read(17,'(4i6,f8.3)') ngrid,gx,gy,gz,gdia
 c
 c        --- now, we can allocate memory:
 c
@@ -279,7 +283,10 @@
          call timer_start(TIME_rdcrd)
          CALL GETCOR(9,NR,X(LCRD),X(LVEL),X(LFORCE),
      $        NTX,BOX,IREST,T)
+         CALL GETGRID(natom,ngrid,x(lgst))
          if( igb.eq.0 .and. induced.eq.1 ) call get_dips(X,nr)
```

```
+c     ** Initialise adding particle stuff
+         if(ap.eq.1) call init_grow(X(LGST),NGRID)
 C
 C        ----- SET THE INITIAL VELOCITIES -----
 C
@@ -502,7 +509,12 @@
 c
 C    ---------------- Old parallel for minimization ----------------------
 c
-      if (imin.ne.0 .or. plevel.eq.0) then
+      if(imin.eq.2) then
+         idumm = 0
+      else
+         idumm = imin
+      endif
+      if (idumm.ne.0 .or. plevel.eq.0) then
         mpi_orig = .true.
         notdone = 1
       else
@@ -586,6 +598,16 @@
         endif
 c

+      elseif (IMIN.eq.2) then
+
+c     ** FILL THE GRID
+
+         call fillgrid(x,ix,ih,ipairs,X(LCRD),X(LWINV),X(Lmass)
+     $        ,X(LFORCE),X(Lvel),X(Lvel2),X(L45),X(Lcrdr),X(L50),X(L95
+     $        ),IX(I70),X(L75),erstop,r_stack,i_stack,X(LGST))
+
+         if (master) CALL MINRIT(X(LCRD),X(LGST),NGRID)
+
       ELSE
 C
 c (IMIN=1)
@@ -600,7 +622,7 @@
 c
 C           --- Write the restart file:
 c
-         if (master) CALL MINRIT(X(LCRD))
+         if (master) CALL MINRIT(X(LCRD),X(LGST),NGRID)
 C
       ENDIF
 c
--- sander_fluid/set.f
```

```
+++ sander_fluid/set.f
@@ -330,6 +330,7 @@
 #endif
 #include "extra.h"
 #include "nmr.h"
+#include "grid.h"
       integer target,fraction
       integer thismol, thismolsfirstnode, thismolslastnode
       integer thismolslastatom, group_world, group_temp
@@ -359,9 +360,11 @@
 c   --- do not stop before or after a single-atom resiude, since this might
 c       be a terminating hydrogen that would have a bond to another residue:
 c
+        if(ipres(ires).lt.gstart.AND.ipres(ires).ge.gstart+gx*gy*gz)then
             if ((ipres(ires+1) - ipres(ires)) .lt. 2) go to 10
             if ( ires.gt.1 .and.
      +          (ipres(ires) - ipres(ires-1)) .lt. 2) go to 10
+        endif
 c
 c   --- if this residue starts past the target atom, end the atom list
 c       at the end of the previous residue:
@@ -371,6 +374,8 @@
               go to 20
             end if
    10     continue
+          if(master) print*,node,nres,ires,ipres(ires),gstart,gstart+gx
+     $         *gy*gz
           write(6,*) 'Error in setpar: check code, input'
           call mexit(6,1)
    20   continue
--- Makefile
+++ Makefile
@@ -5,6 +5,8 @@
  cd lib; make install
  cd addles; make install
  cd sander; make install
+ cd sander_grid; make install
+ cd sander_fluid; make install
  cd ptraj; make install
  cd carnal; make install
  cd gibbs; make install
@@ -39,8 +41,10 @@
 install.parallel:
  -mkdir ../exe
  cd sander; make install
+ cd sander_grid; make install
```

```
+ cd sander_fluid; make install
  cd gibbs; make install
-
+
 clean::
 # rm -f MACHINE
  cd lib; make clean
@@ -49,6 +53,8 @@
  cd arpack; make clean
  cd addles; make clean
  cd sander; make clean
+ cd sander_grid; make clean
+ cd sander_fluid; make clean
  cd ptraj; make clean
  cd carnal; make clean
  cd gibbs; make clean
```