

Case Study 1: Equation of State of the Lennard-Jones Fluid: $P(N,V,T)$

- NVT MC simulation of LJ atoms, no external forces

- interaction: LJ

$$u(r) = \begin{cases} u^{lj}(r) & r \leq r_c \\ 0 & r > r_c, \end{cases}$$

- the pressure P is calculated by the Virial:

$$P = \frac{\rho}{\beta} + \frac{\text{vir}}{V}, \quad \text{vir} = \frac{1}{3} \sum_i \sum_{j>i} \mathbf{f}(\mathbf{r}_{ij}) \cdot \mathbf{r}_{ij},$$

- Truncation: simple, $r_c = L/2$
- Tail corrections for U and P (only 3D bulk)

$$u^{\text{tail}} = \frac{8}{3}\pi\rho \left[\frac{1}{3} \left(\frac{1}{r_c} \right)^9 - \left(\frac{1}{r_c} \right)^3 \right]$$

$$p^{\text{tail}} = \frac{16}{3}\pi\rho^2 \left[\frac{2}{3} \left(\frac{1}{r_c} \right)^9 - \left(\frac{1}{r_c} \right)^3 \right].$$

Basic Algorithm

<pre>PROGRAM mc</pre>	basic Metropolis algorithm
<pre>do icycl=1,ncycl</pre>	perform <code>ncycl</code> MC cycles
<pre> call mcmove</pre>	displace a particle
<pre> if (mod(icycl,nsamp).eq.0)</pre>	
<pre>+ call sample</pre>	sample averages
<pre> enddo</pre>	
<pre>end</pre>	

- Subroutine **mcmove** attempts to displace a randomly selected particle
- Subroutine **sample** samples quantities every `nsamp` cycle.

Subroutine **mcmove**: Attempt to Displace a Particle

<pre>SUBROUTINE mcmove o=int(ranf()*npart)+1 call ener(x(o),eno) xn=x(o)+(ranf()-0.5)*delx call ener(xn,enn) if (ranf().lt.exp(-beta + *(enn-eno)) x(o)=xn return end</pre>	<p>attempts to displace a particle</p> <p>select a particle at random energy old configuration give particle random displacement energy new configuration acceptance rule (3.2.1) accepted: replace $x(o)$ by xn</p>
--	--

- Subroutine **ener** calculates the energy of a particle at the given position.
- Note that, if a configuration is rejected, the old configuration is retained.
- The **ranf** () is function that generates a random number uniform in [0,1]

Program structure of the simulation: reads
Main: `mc_nvt.f` → input, initializes, does the loop,
write output, etc

Subroutines:

`adjust.f` → adjusts parameter Δ

`coru.f` → calculates tail correction to the energy

`eneri.f` → calculates the energy and virial of one particle

`ranf.f` → RNG

`readdat.f` → reads input data

store.f → writes configuration

corp.f → calculates tail correction to pressure

ener.f → calculates the LJ interaction between two particles

lattice.f → set initial configuration = all particles in a square lattice

mcmove.f → attemps to move a particle

rantest.f → test the RNG at the beginning

sample.f → writes results of average energy, pressure

toterg.f → calculates the total energy and virial of the system

PROGRAM MC_NVT

MAIN PROGRAM MC_NVT

```
C-----
C
C   Understanding Molecular Simulations: From Algorithms to Applications
C
C           Daan Frenkel   and   Berend Smit
C
C   We make no warranties, express or implied, that the programs contained
C   in this work are free of error, or that they will meet your requirements
C   for any particular application. They should not be relied on for solving
C   problems whose incorrect solution could result in injury, damage, or
C   loss of property. The authors and publishers disclaim all liability for
C   direct or consequential damages resulting from your use of the programs
C
C-----
C
C
C   Case Study 1: Equation of state of the Lennard-Jones fluid
C
C-----
```

c... comments

```
IMPLICIT NONE
INTEGER iseed, equil, prod, nsamp, ii, icycl, ndispl, attempt,
&      nacc, ncycl, nmoves, imove
DOUBLE PRECISION en, ent, vir, virt, dr
```

Declaration of types of
variables, always first

DEFAULT: REAL (A-H,O-Z), INTEGER (I-M)

```

IMPLICIT NONE
INTEGER iseed, equil, prod, nsamp, ii, icycl, ndispl, attempt,
&      nacc, ncycl, nmoves, imove
DOUBLE PRECISION en, ent, vir, virt, dr

```

suroutine **READDAT**: reads input data, calculates parameters, initialization (fort.15, fort.25)

```

WRITE (6, *) '***** MC_NVT *****'
c ---initialize sysem
CALL READDAT(equil, prod, nsamp, ndispl, dr, iseed)
nmoves = ndispl

```

```

c ---total energy of the system
CALL TOTERG(en, vir)

```

subroutine **TOTERG** calculates the total energy and virial, tail corrections.

```

WRITE (6, 99001) en, vir
c ---start MC-cycle

```

```

DO ii = 1, 2

```

```

c --- ii=1 equilibration

```

```

c --- ii=2 production

```

first cycles are for equilibration: MC steps, NO averages. # = equil
last cycles are for production: MC steps AND averages. # = prod

```

IF (ii.EQ.1) THEN

```

```

    ncycl = equil

```

```

    IF (ncycl.NE.0) WRITE (6, *) 'Start equilibration'

```

```

ELSE

```

```

    IF (ncycl.NE.0) WRITE (6, *) 'Start production'

```

```

    ncycl = prod

```

```

END IF

```

```

attempt = 0

```

initialization

```

nacc = 0

```

```

c ---intialize the subroutine that adjust

```

subroutine **ADJUST**: change the value of dr (Δ) to keep good acceptance rate.

```

CALL ADJUST(attempt, nacc, dr)

```



```

DO icycl = 1, ncycl
  DO imove = 1, nmoves
    ---attempt to displace a particle
    CALL MCMOVE(en, vir, attempt, nacc, dr, iseed)
  END DO
  IF (ii.EQ.2) THEN
    ---sample averages
    IF (MOD(icycl,nsamp).EQ.0) CALL SAMPLE(icycl, en, vir)
  END IF
  IF (MOD(icycl,ncycl/5).EQ.0) THEN
    WRITE (6, *) '=====>> Done ', icycl
    ---write intermediate configuration
    CALL STORE(8, dr)
    ---adjust maximum displacements
    CALL ADJUST(attempt, nacc, dr)
  END IF
END DO
IF (ncycl.NE.0) THEN
  IF (attempt.NE.0) WRITE (6, 99003) attempt, nacc,
    100.*FLOAT(nacc)/FLOAT(attempt)
  ---test total energy
  CALL TOTERG(ent, virt)
  IF (ABS(ent-en).GT.1.D-6) THEN
    WRITE (6, *)
    ' ##### PROBLEMS ENERGY ##### '
  END IF

```

MC loop: # cycles = ncycl

nmoves in each cycle

subroutine **MCMOVE**: attempts and accepts/rejects move

production cycle:
subroutine **SAMPLE**
calculates and writes averages
every nsample moves.

end MC loop

test energy

```

        END IF
        IF (ABS(virt-vir).GT.1.D-6) THEN
            WRITE (6, *)
&          ' ##### PROBLEMS VIRIAL ##### '
        END IF
        WRITE (6, 99002) ent, en, ent - en, virt, vir, virt - vir
    END IF
END DO
CALL STORE(21, dr)
STOP

```

test virial

save, print

```

99001 FORMAT (' Total energy initial configuration: ', f12.5, /,
&          ' Total virial initial configuration: ', f12.5)
99002 FORMAT (' Total energy end of simulation      : ', f12.5, /,
&          '          running energy              : ', f12.5, /,
&          '          difference                   : ', e12.5, /,
&          ' Total virial end of simulation        : ', f12.5, /,
&          '          running virial               : ', f12.5, /,
&          '          difference                   : ', e12.5)
99003 FORMAT (' Number of att. to displ. a part.  : ', i10, /,
&          ' success: ', i10, '(= ', f5.2, '%)')
END

```

SUBROUTINE MCMOVE

SUBROUTINE MCMOVE(En, Vir, Attempt, Nacc, Dr, Iseed)

attempts to displace

variables will be pass to / from caller

Ener (input/output) : total energy

Vir (input/output) : total virial

Attemp (input/output) number of attemps that have been
performed to displace a particle

Nacc (input/output) number of successful attemps
to displace a particle

Dr (input) maximum displacement

Iseed (input) seed random number (not used in present
random number generator)

IMPLICIT NONE

INCLUDE 'parameter.inc'

INCLUDE 'conf.inc'

INCLUDE 'system.inc'

DOUBLE PRECISION enn, eno, En, RANF, xn, yn, zn, viro, virn, Vir,
& Dr

INTEGER o, Attempt, Nacc, jb, Iseed

SUBROUTINE MCMOVE

SUBROUTINE MCMOVE(En, Vir, Attempt, Nacc, Dr, Iseed)

attempts to displace

variables will be pass to / from caller

Ener (input/output) : total energy

Vir (input/output) : total virial

Attemp (input/output) number of attemps that have been
performed to displace a particle

Nacc (input/output) number
to displace

Dr (input) maximum displacement

Iseed (input) seed random
random number

c parameters.inc

integer npmax

parameter (npmax=10000)

c

c npmax : maximum number of particles

conf.inc

double precision x,y,z

integer npart

common/conf1/x(npmax),y(npmax),z(npmax),npart

system.inc

double precision box,temp,beta,hbox

common/sys1/box,hbox,temp,beta

IMPLICIT NONE

INCLUDE 'parameter.inc'

INCLUDE 'conf.inc'

INCLUDE 'system.inc'

DOUBLE PRECISION enn, end

& Dr

INTEGER o, Attempt, Nacc

```
Attempt = Attempt + 1
```

counts # attemps

```
jb = 1
```

```
c ---select a particle at random
```

```
o = INT(NPART*RANF(Iseed)) + 1
```

RNG

Subroutine **ENERI**
calculates the energy
of one particle

```
c ---calculate energy old configuration
```

```
CALL ENERI(X(o), Y(o), Z(o), o, jb, eno, viro)
```

```
c ---give particle a random displacement
```

```
xn = X(o) + (RANF(Iseed)-0.5D0)*Dr
```

```
yn = Y(o) + (RANF(Iseed)-0.5D0)*Dr
```

```
zn = Z(o) + (RANF(Iseed)-0.5D0)*Dr
```

energy
eno= before move
enn= after move

```
c ---calculate energy new configuration:
```

```
CALL ENERI(xn, yn, zn, o, jb, enn, virn)
```

```
c ---acceptance test
```

```
IF (RANF(Iseed).LT.EXP(-BETA*(enn-eno))) THEN
```

```
c --accepted
```

acceptance rule

if accepted

```
  Nacc = Nacc + 1
```

```
  En = En + (enn-eno)
```

```
  Vir = Vir + (virn-viro)
```

calculate new energy and virial

c

if accepted

---put particle in simulation box

```
IF (xn.LT.0) xn = xn + BOX
IF (xn.GT.BOX) xn = xn - BOX
IF (yn.LT.0) yn = yn + BOX
IF (yn.GT.BOX) yn = yn - BOX
IF (zn.LT.0) zn = zn + BOX
IF (zn.GT.BOX) zn = zn - BOX
X(o) = xn
Y(o) = yn
Z(o) = zn
END IF
RETURN
END
```

use periodic boundary conditions
replace the particle position with the new one

SUBROUTINE ENERI

in

out

SUBROUTINE ENERI(Xi, Yi, Zi, I, Jb, En, Vir)

```
c
c   calculates the energy of particle I with particles j=jb,npart
c
c   Xi (input) x coordinate particle I
c   Yi (input) y coordinate particle I
c   Zi (input) z coordinate particle I
c   I  (input) particle number
c   Jb (input) = 0 calculates energy particle I with all other particle
c               = jb calculates energy particle I with all particles j > jb
c   En  (output) energy particle i
c   Vir (output) virial particle i
c
c
c   IMPLICIT NONE
c   INCLUDE 'parameter.inc'
c   INCLUDE 'conf.inc'
c   INCLUDE 'system.inc'
c
c   DOUBLE PRECISION Xi, Yi, Zi, En, dx, dy, dz, r2, Vir, virij, enij
c   INTEGER I, j, Jb
```

c


```


En = 0
Vir = 0
DO j = Jb, NPART
  IF (j.NE.I) THEN
    dx = Xi - X(j)
    dy = Yi - Y(j)
    dz = Zi - Z(j)
    IF (dx.GT.HBOX) THEN
      dx = dx - BOX
    ELSE
      IF (dx.LT.-HBOX) dx = dx + BOX
    END IF
    IF (dy.GT.HBOX) THEN
      dy = dy - BOX
    ELSE
      IF (dy.LT.-HBOX) dy = dy + BOX
    END IF
    IF (dz.GT.HBOX) THEN
      dz = dz - BOX
    ELSE
      IF (dz.LT.-HBOX) dz = dz + BOX
    END IF
    r2 = dx*dx + dy*dy + dz*dz
    CALL ENER(enij, virij, r2)
    En = En + enij
    Vir = Vir + virij
  END IF
END DO
RETURN
END

```

Sum over all other particles

Find the nearest image

out in

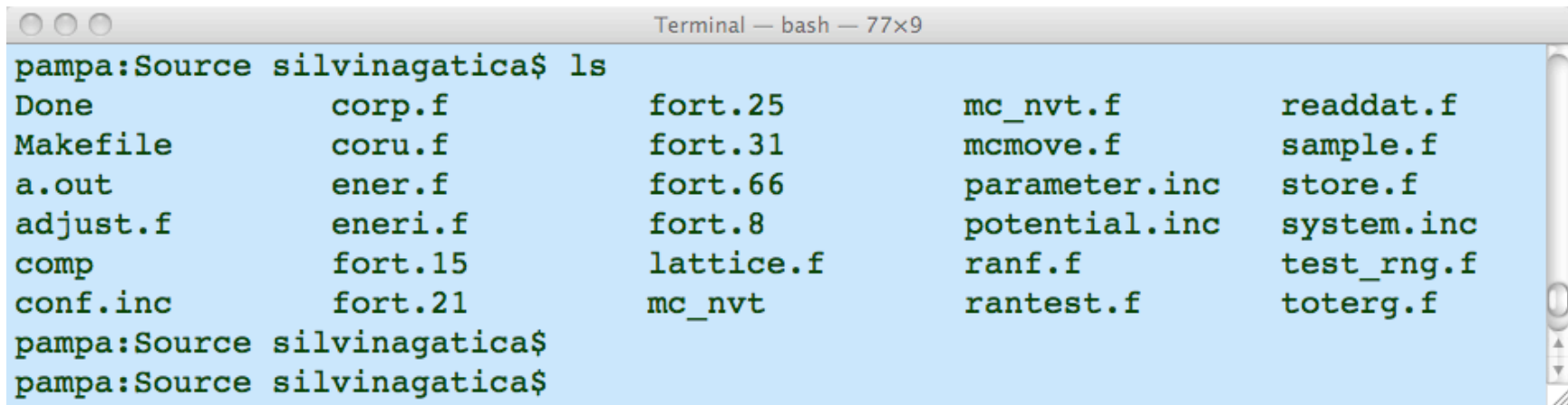


Subroutine **ENER(enij,virij,r2)** calculates the LJ interaction and virial between particles i,j with cutoff rc

HOW TO RUN

In UNIX/Linux

In MAC: open application TERMINAL



```
Terminal — bash — 77x9
pampa:Source silvinagatica$ ls
Done                corp.f             fort.25            mc_nvt.f           readdat.f
Makefile            coru.f             fort.31            mcmove.f           sample.f
a.out               ener.f             fort.66            parameter.inc       store.f
adjust.f            eneri.f            fort.8             potential.inc       system.inc
comp                fort.15            lattice.f          ranf.f             test_rng.f
conf.inc            fort.21            mc_nvt             rantest.f          toterg.f
pampa:Source silvinagatica$
pampa:Source silvinagatica$
```

DO NOT MODIFY THE PROGRAMS

1. COMPILE the program:

`ifort mc_nvt.f adjust.f corp.f coru.f ener.f eneri.f lattice.f mcmove.f ranf.f ... rantest.f readdat.f sample`

- Replace ifort with your compiler (f77, g77)
- An executable “a.out” will be created.

2. Create the data files fort.15 and fort.25 in the same directory where a.out is. Use any text editor without format (vi, emacs, TextEdit...)
3. Run: write the command:
`./a.out`
4. After the program stop there will be new files created during run: `fort.66` contains the energy and pressure
5. To see the file write:
`more fort.66`