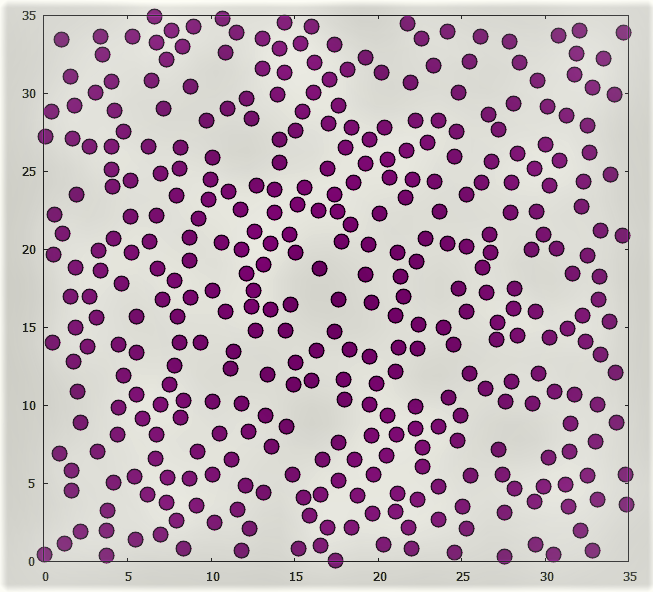
***Wykonał:***

***Rafał Świętek***

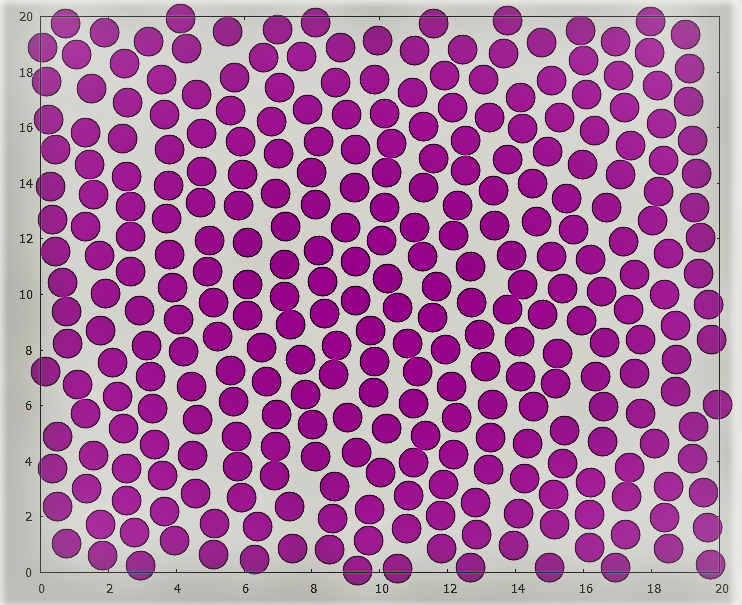
***Nr indeksu: 236668***

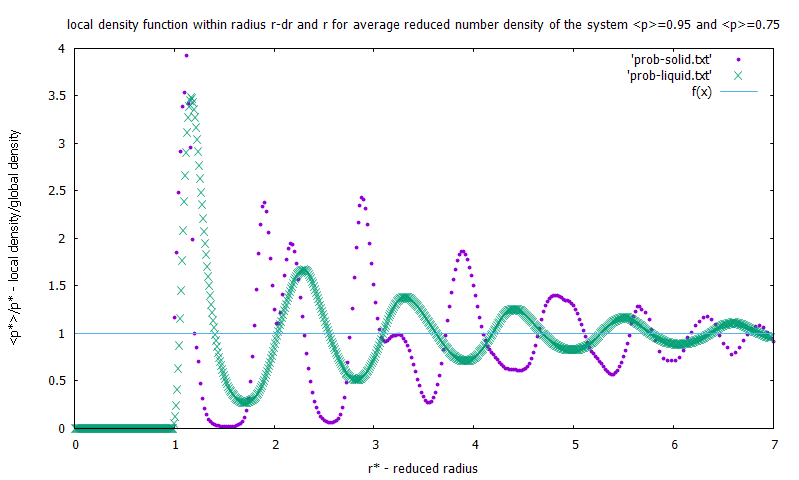
***Two body correlation in 2D Lennard- Jones liquid***

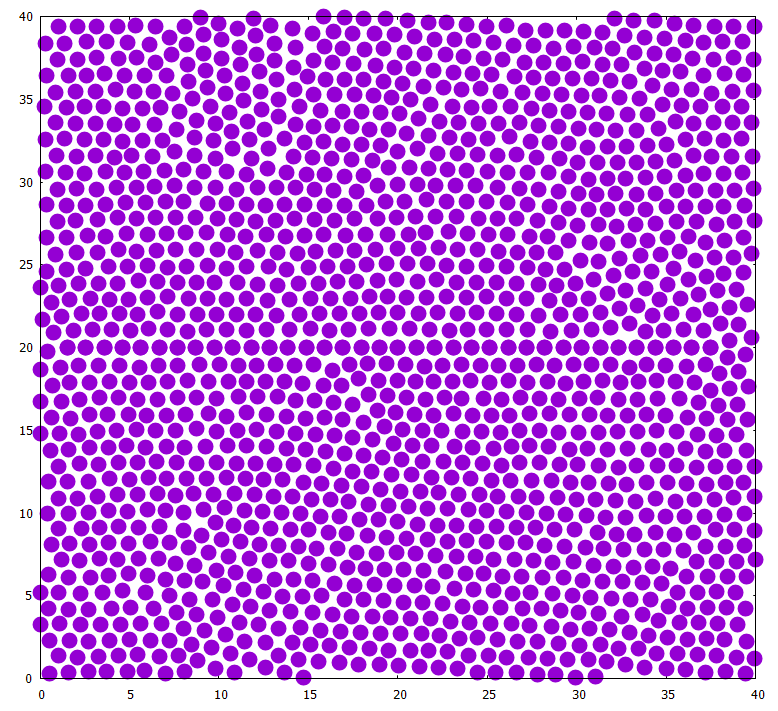
Symulacja polegała na analizie zachowania określonej liczby cząstek (określona koncentracja) w zamkniętej przestrzeni o wymiarach , gdzie to przekrój czynny cząstki. W symulacji zdefiniowano wielkości zredukowane Takich jak:

 Przeprowadzono w temperaturze zredukowanej . Zauważono konfiguracje dla różnej zredukowanej koncentracji układu takich jak:

Poniższą konfigurację wykonano dla układu o wymiarach







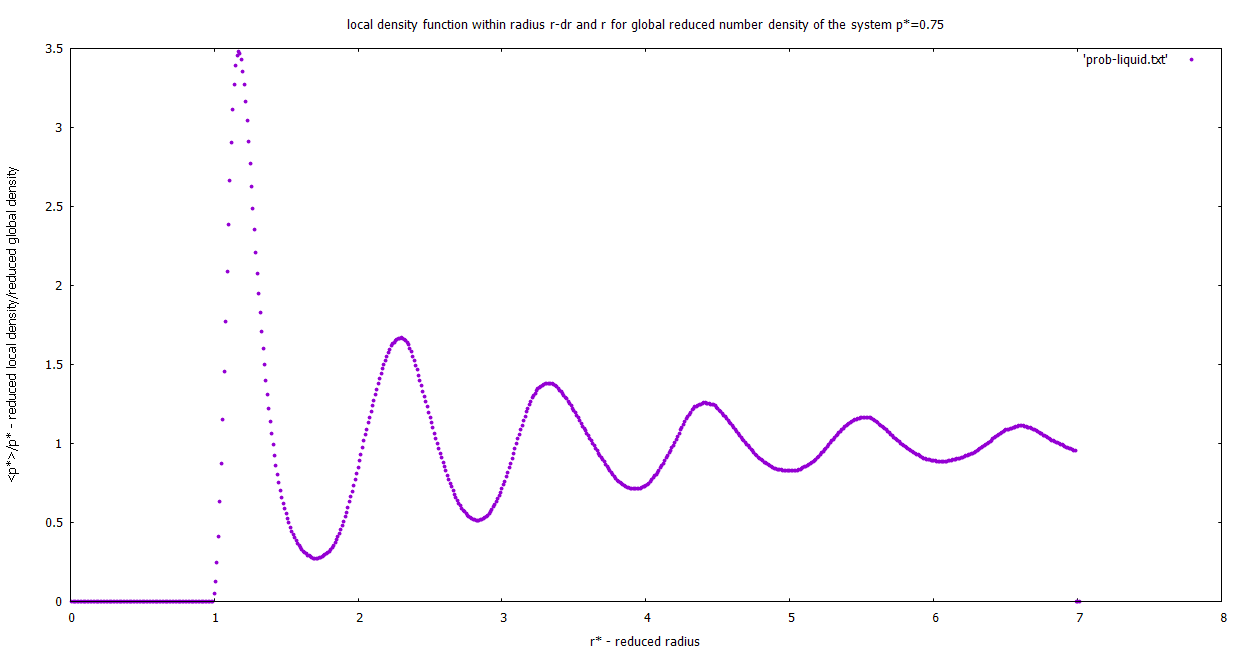
Jak widzimy dla wysokich koncentracji układ dąży do ustawienia się w sieć trójkątną, gdyż jest najbardziej optymalna energetycznie. Powyższą konfigurację narysowano dla układu kwadratowego o boku równym

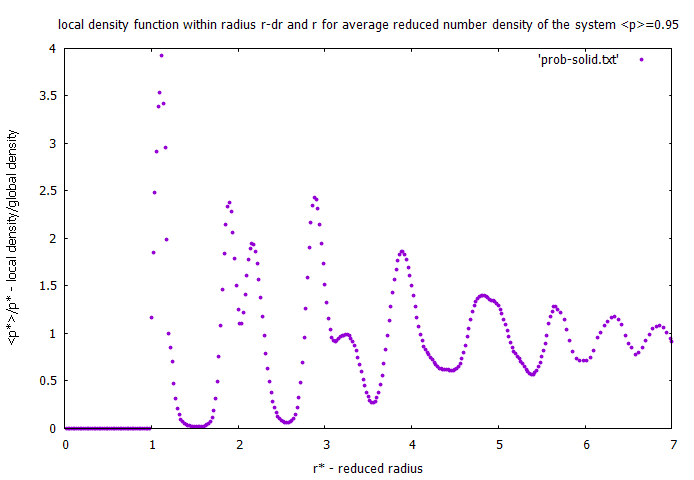
Następnie przeliczono lokalna gęstość układu w pierścieniu o promieniach r-dr i r, po czym uśredniono po wszystkich cząstkach (czyli średnia lokalna gęstość).

Układ badany miał wymiary i badano go w temperaturze ,

dla koncentracji oraz . Wykresy znajdują się na kolejnej stronie:

Na wykresie oznaczono niebieską linia maksimum, które oznaczono przez , gdzie promień (na osi x) jest zmienną zredukowana postaci . Przedstawiono wykres stosunku średniej gęstości lokalnej do gęstości globalnej w zależności od promienia zewnętrznego pierścienia. Na poprzedniej stronie obie krzywe zostały przedstawione na jednym wykresie w celach porównawczych.





program LennardJonesLiquid

       implicit none

       integer L,N,MCS,i,j,k,d, sim\_time, check\_time, m

       parameter (L=20, MCS=230000, sim\_time = 30000, check\_time = 100)

       real r\_c, x\_step, y\_step,r\_step, ran1, P, dr, ro, PI, T

       parameter (r\_c = 2.5, x\_step = 0.01,y\_step = 0.01,r\_step = 0.02,

     &                  dr = 0.05 ,PI = 3.141592653, ro = 0.95, T = 0.7)

*! 0.8 - liquid & 0.93 - solid*

       real xnew, ynew, dx, dy, r\_ij2, Rnew2, dU, r, Probability, Rmax

       real dx\_new, dy\_new

       integer counter, r\_size, mcs\_size

       real, allocatable:: x(:), y(:), Prob(:,:)

       d =-1

       Rmax = L/3.0 - 1.0

*!--------*

       r\_size = int( Rmax/r\_step + 1.0 )

       mcs\_size = int( (mcs-sim\_time)/check\_time + 1.0)

*!---------- matrix places*

       N = int( ro\*L\*\*2 + 0.0)

       write(\*,\*) 'N = ', N

       allocate (x(N), y(N), Prob(r\_size,mcs\_size) )

       call Liquid(L,N,x,y)

       do i=1, r\_size

         do j=1, mcs\_size

            Prob(i,j) = 0.0

         enddo

       enddo

       call showmatrix(N,x,y)

       write(\*,\*) 'MCS'

        counter = 1

        do k=1, MCS

          do i=1, N

             dU = 0

             xnew = x(i) + (ran1(d)-0.5)\*x\_step

             ynew = y(i) + (ran1(d)-0.5)\*y\_step

             if(xnew.gt.L) xnew = xnew - L

             if(ynew.gt.L) ynew = ynew - L

             if(xnew.lt.0.) xnew = xnew + L

             if(ynew.lt.0.) ynew = ynew + L

              do j=1, N

               if(j.ne.i) then

                  dx = abs( x(i) - x(j) )

                  dy = abs( y(i) - y(j) )

                  if(dx.gt.L/2.) dx = L - dx

                  if(dy.gt.L/2.) dy = L - dy

                  r\_ij2 = dx\*\*2 + dy\*\*2

                  dx\_new = abs( xnew-x(j) )

                  dy\_new = abs( ynew-y(j) )

                  if(dx\_new.gt.L/2.) dx\_new = L - dx\_new

                  if(dy\_new.gt.L/2.) dy\_new = L - dy\_new

                  Rnew2 = dx\_new\*\*2 + dy\_new\*\*2

                  if(Rnew2.le.1.0) goto 7

*!r^2<=1 then r<=1, sqrt() not needed*

                  if(r\_ij2.le.r\_c.and.Rnew2.le.r\_c) then

                     dU = dU + 4.0\*(1/Rnew2\*\*6 - 1/r\_ij2\*\*6)

                  endif

               endif

              enddo

              if(ran1(d).le.min(1.0,exp(-dU/T))) then

                  x(i) = xnew

                  y(i) = ynew

              endif

7           continue

          enddo

          if(k.ge.sim\_time.and.mod(k,check\_time).eq.0) then

             r = 1.0

             m = INT(1/r\_step - 1.0)

33           continue

                Prob(m,counter) = Prob(m,counter) +

     &                   Probability(r,dr,N,L,ro,x,y) *!probabilistic matrix*

                r = r + r\_step

                m = m + 1

             if(r.le.Rmax) goto 33

             counter = counter + 1

          endif

          write(\*,\*) k

        enddo

        write(\*,\*) '----------------------------------'

        write(\*,\*) 'Probability', '     ', 'r'

        call save\_prob(Prob,r\_size,mcs\_size,r\_step)

        call showmatrix(N,x,y)

        pause

*!call showmatrix(N,x,y)*

       end program

function Probability(r,dr,N,L,ro,x,y)

        integer i, N, j, L

        real dr, deltaR2, r, x(N), y(N)

        real P, Probability, PI, ro, dx, dy

        PI = 3.141592653

        P = 0

        Probability = P

        if(r.le.dr) return

        do j=1, N

          M = 0

          do i=1, N

          dx = abs( x(i) - x(j) )

          dy = abs( y(i) - y(j) )

          if(dx.gt.L/2.) dx = L - dx

          if(dy.gt.L/2.) dy = L - dy

          deltaR2 = sqrt( dx\*\*2 + dy\*\*2 )

          if(deltaR2.gt.r-dr.and.deltaR2.le.r) then

                M = M + 1

          endif

          enddo

          P = P + M/(2.0\*PI\*r\*dr\*ro)

*!ring width: r-dr to r*

        enddo

        Probability = P/(N+0.0)

        return

        end

subroutine Liquid(L,N,x,y)

        integer L, N, M, k

        real x(N), y(N), d, free\_space

        x(1) = 0.5

        y(1) = 0.5

        M = (L-1)\*(L-2)

        k = int(sqrt(N+0.0))

        free\_space = (L-1.0)/(k+0.0) - 1.0

        if(N.ge.M) then

           d = 1.0

        else

           d = 1.0 + free\_space

        endif

        write (\*,\*) 'd = ',d

        do i=2, N

          y(i) = y(i-1)

          x(i) = x(i-1) + d

          if(x(i).gt.L-0.5) then

             y(i) = y(i) + d

             x(i) = 0.5

          endif

        enddo

       end

       subroutine showmatrix(n,x,y)

       integer n

       real x(n), y(n)

       open(99,file='matrix-0.95.txt')

       do i=1, n

          write(99,\*) x(i), y(i)

       enddo

       close(99)

       end

       subroutine save\_prob(Prob, A, B, dr)

        integer A,B, i, j

        real Prob(A,B), P

        open(13,file='prob-solid.txt')

        do i=1, A

           P = 0.0

           do j=1, B

              P = P + Prob(i,j)

           enddo

           P = P/(B+0.0)

           write(13,\*) i\*dr, P

           write(\*,\*) i\*dr, P

        enddo

        close(13)

       end

FUNCTION ran1(idum)

      INTEGER idum,IA,IM,IQ,IR,NTAB,NDIV

      REAL ran1,AM,EPS,RNMX

      PARAMETER (IA=16807,IM=2147483647,AM=1./IM,IQ=127773,IR=2836,

     \*NTAB=32,NDIV=1+(IM-1)/NTAB,EPS=1.2e-7,RNMX=1.-EPS)

      INTEGER j,k,iv(NTAB),iy

      SAVE iv,iy

      DATA iv /NTAB\*0/, iy /0/

      if (idum.le.0.or.iy.eq.0) then

        idum=max(-idum,1)

        do 11 j=NTAB+8,1,-1

          k=idum/IQ

          idum=IA\*(idum-k\*IQ)-IR\*k

          if (idum.lt.0) idum=idum+IM

          if (j.le.NTAB) iv(j)=idum

11      continue

        iy=iv(1)

      endif

      k=idum/IQ

      idum=IA\*(idum-k\*IQ)-IR\*k

      if (idum.lt.0) idum=idum+IM

      j=1+iy/NDIV

      iy=iv(j)

      iv(j)=idum

      ran1=min(AM\*iy,RNMX)

      return

      END