program LiquidCristal

       implicit none

       integer MCS, L, D, i, j, k, p, a, b

       real PI, Dfi, dU, ksi, n\_e, n\_o, tol, E0, kappa, omega

       parameter (MCS=230000, D=20, L=50, PI=3.141592653)

       parameter (ksi=20, n\_e=1.7, n\_o=1.5)

       parameter (kappa = 1666000, omega = 12e15\*PI)

*!lambda ~ 600nm and f~ 6\*10^14Hz -fala monochromatyczna płaska*

*!parameter (kappa = 20., omega = 5\*PI.) - mikrofale*

*!parameter (kappa = 2., omega = 120\*PI.)  - fale ultrakrotkie*

*!D - const szerokość: Dz*

       real P2, n, w, H

       real psi, ran1, R, refractionindex, Hamiltonian

*!psi - finew*

       real, allocatable:: fi(:,:), E(:,:,:)

       integer, allocatable::  niz(:), piz(:), nix(:), pix(:)

       allocate (niz(D), piz(D), nix(L), pix(L))

       allocate (fi(L,D), E(L,D,MCS))

       do i=1, L

        do j=1, D

           fi(i,j) = 0

        enddo

       enddo

       open(1,file='n(E)L=50-neww-k.txt')

       open(2,file='EnergyL=50-neww-k.txt')

       p = -1

       call neighbour(L,nix,pix)

       call neighbour(D,niz,piz)

       call Electricfield(L,D,MCS,E,kappa,omega)

*!additional for monochromatic wave*

       E0 = 0

       Dfi = PI/18.

11     continue

***Wykonał:***

***Rafał Świętek***

***Nr indeksu: 236668***

        a = 0

        n = 0

        H = 0

        do k=1, MCS

          do i=1, L

             do j=2, D-1

                R = ran1(p)

                psi = fi(i,j) + (R-0.5)\*Dfi

                if(psi.gt.PI/2.) psi = psi - PI

                if(psi.lt.-PI/2.) psi = psi + PI

  dU = sin(fi(i,j)+psi-2\*fi(pix(i),j))\*sin(psi-fi(i,j)) +

     &        sin(fi(i,j)+psi-2\*fi(nix(i),j))\*sin(psi-fi(i,j)) +

     &        sin(fi(i,j)+psi 2\*fi(i,piz(j)))\*sin(psi-fi(i,j)) +

     &        sin(fi(i,j)+psi-2\*fi(i,niz(j)))\*sin(psi-fi(i,j))

                dU = dU\*ksi\*1.5

                dU=dU-1.5\*sin(psi+fi(i,j))\*sin(psi-fi(i,j))\*

     &                (E0\*\*2)\*(E(i,j,k))\*\*2

                w = min(1.0,exp(-dU))

                if(ran1(p).le.w) fi(i,j) = psi

             enddo

          enddo

          if(k.ge.30000.and.mod(k,100).eq.0) then

             a = a + 1

             n = n + refractionindex(L,D,fi,n\_e,n\_o)

             H = H + Hamiltonian(L,D,MCS,fi,nix,niz,E,E0,ksi,k)

          endif

        enddo

       n = n/a

       H = H/a

       write(1,\*) E0, n

       write(2,\*) E0, H

       write(\*,\*) E0, n, H

       call increment(E0, 0.95, 1.15, 0.01)

*!within a to b increment is 0.005*

*!E0 = E0 + 0.1*

       if(E0.le.5) goto 11

       call showmatrix(L,D,fi)

       close(1)

       end

subroutine neighbour(k,next, prev)

integer k, next(k), prev(k)

        do i=1, k

          next(i) = i+1

          prev(i) = i-1

        enddo

        next(k)=1

        prev(1)=k

       end

       function P2(x)

       real x, P2

       P2 = 1.5\*( cos(x) )\*\*2 - 0.5

       return

       end

\*       funkcja cosinus jest PARZYSTA*!!*

       subroutine increment(E,a,b,dE)

        real E, a, b, dE

        if(E.ge.a.and.E.le.b) then

           E = E + dE

        else

           E = E + 0.05

        endif

       end

       subroutine Electricfield(Dx,Dz,mcs,Epola,k,omega)

       integer Dx, Dz, x, z, t

real Epola(Dx,Dz, mcs), r, k, omega

       do t=1, mcs

        do x=1, Dx

          do z=1, Dz

             r = sqrt( x\*\*2 + z\*\*2 + 0.0 )

             Epola(x,z,t) = cos(k\*r - omega\*t)

          enddo

        enddo

        write(\*,\*) t, k, omega

       enddo

       end

       function Hamiltonian(Dx,Dz,mcs,fi,nx,nz,Epola,E\_0,ksi,m)

       integer Dx, Dz, nx(Dx), nz(Dz), i, j, N

       real fi(Dx,Dz), H, Hamiltonian

       real P2, U, PI, ksi

       real Epola(Dx,Dz,mcs)

       PI = 3.141592653

       H = 0

       do i=1, Dx

         do j=1, Dz-1

            U=ksi\*( P2(fi(i,j)-fi(nx(i),j)) + P2(fi(i,j)-fi(i,nz(j))) )

            U = U + P2(PI/2.-fifi(i,j))\*(E\_0\*\*2)\*(Epola(i,j,m))\*\*2

            H = H - U

         enddo

         H = H - P2(fi(i,Dz)-fi(nx(i),Dz))

         H = H - P2(PI/2.-fi(i,Dz))\*(E\_0\*\*2)\*(Epola(i,Dz,m))\*\*2

       enddo

       N = Dx\*Dz

       H = H/N

       Hamiltonian = H

       return

       end

       function refractionindex(Dx,Dz,fi,n\_e,n\_o) integer Dx, Dz

        real fi(Dx, Dz), n\_o, n\_e, ne(Dx,Dz), z, y real refractionindex

        y = 0

        do i=1, Dx

           z = 0

           do j=1, Dz

        ne(i,j)=1/sqrt((n\_e/n\_o)\*\*2\*(sin(fi(i,j)))\*\*2+(cos(fi(i,j)))\*\*2)

           z = z + ne(i,j)\*n\_e

           enddo

           y = y + z/Dz

        enddo

        refractionindex = y/Dx

        return

        end

       subroutine showmatrix(Dx, Dz, B)

       integer Dx, Dz

       real B(Dx,Dz)

       open(88,file='ehm.txt')

       do j=1, Dz

          write(88,\*)(B(i,j), i=1, Dx)

       enddo

       close(88)

       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

program Fredericks\_transition

        integer L, k, i

        real Ef, npp1, npp2, npp3, a, b, c, h

*!mpp - 2nd differentiation of n1*

        real m1, m2, m3

        real, allocatable:: n1(:), n2(:), n3(:), Etab(:)

        k = 100

        allocate (n1(k), n2(k), n3(k), Etab(k))

        open(40,file='n(E)L=40.txt')

        open(80,file='n(E)L=80.txt')

        open(120,file='n(E)L=120.txt')

        open(77,file='Ef.txt')

        do i=1,k

         read(40,\*) Etab(i), n1(i)

         read(80,\*) E, n2(i)

         read(120,\*) E, n3(i)

         if(E.gt.1.5) EXIT

         write(\*,\*) Etab(i), n1(i), n2(i), n3(i)

        enddo

        h = 0.01\*\*2

        npp1 = abs(n1(3)+n1(1)-2\*n1(2))/h

        npp2 = abs(n2(3)+n2(1)-2\*n2(2))/h

        npp3 = abs(n3(3)+n3(1)-2\*n3(2))/h

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

*!three-point scheme for evaluating the second derivative*

        do i=3, k-1

          a = abs(n1(i+1)+n1(i-1)-2\*n1(i))/h

          b = abs(n2(i+1)+n2(i-1)-2\*n2(i))/h

          c = abs(n3(i+1)+n3(i-1)-2\*n3(i))/h

          if(a.lt.npp1) then

            npp1 = a

            m1 = Etab(i)

          endif

          if(b.lt.npp2) then

             npp2 = b

             m2 = Etab(i)

          endif

          if(c.lt.npp3) then

             npp3 = c

             m3 = Etab(i)

          endif

          if(Etab(i).gt.1.5) EXIT

        enddo

        Ef = (m1 + m2 + m3)/3

        write(\*,\*) 'E\_fredericks = ' , Ef

*!write(77,\*) Ef*

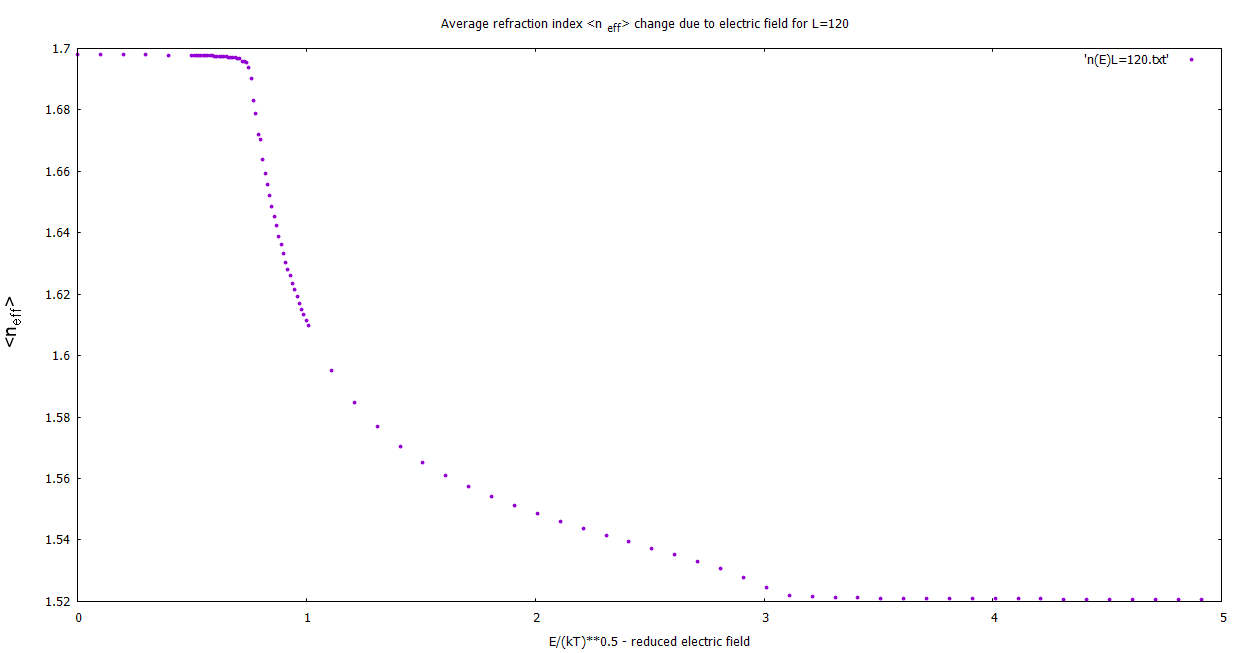
        pause

        end

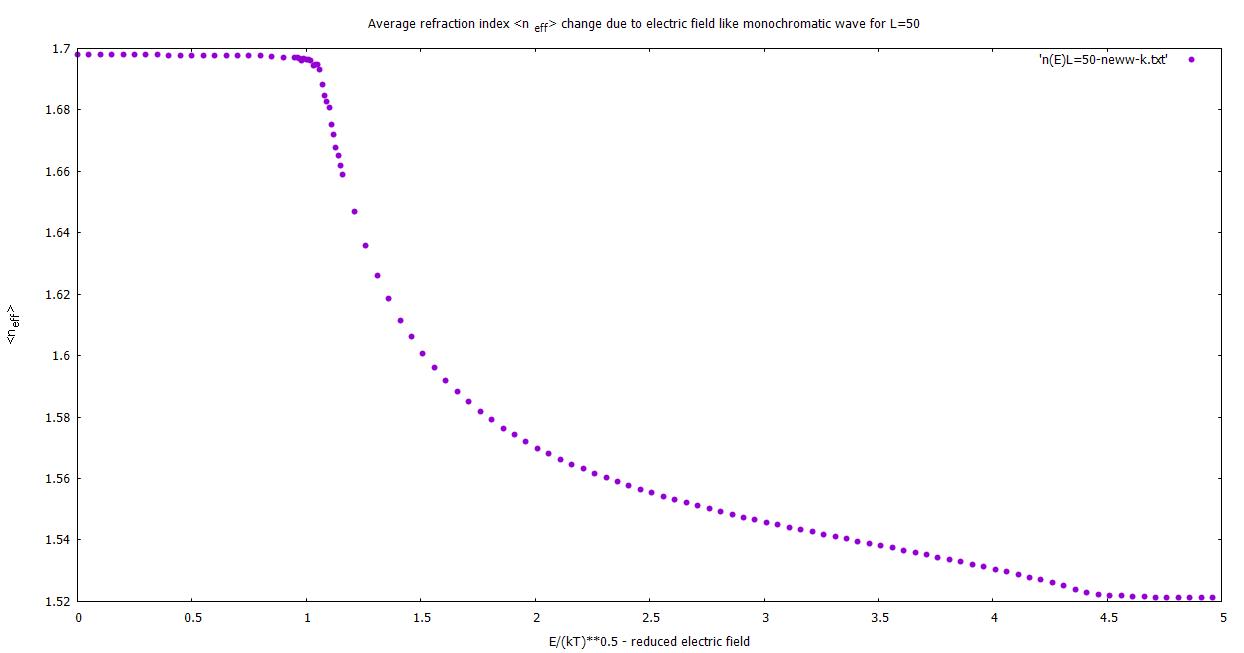
Najpierw przeprowadzono symulację dla jednorodnego pola.

Korzystając z powyższego programu ‘Fredericks transition’ otrzymano próg Freedericksza dla stałej szerokości D=20 równy w przybliżeniu:

Otrzymany wykres przedstawiono na kolejnej stronie.



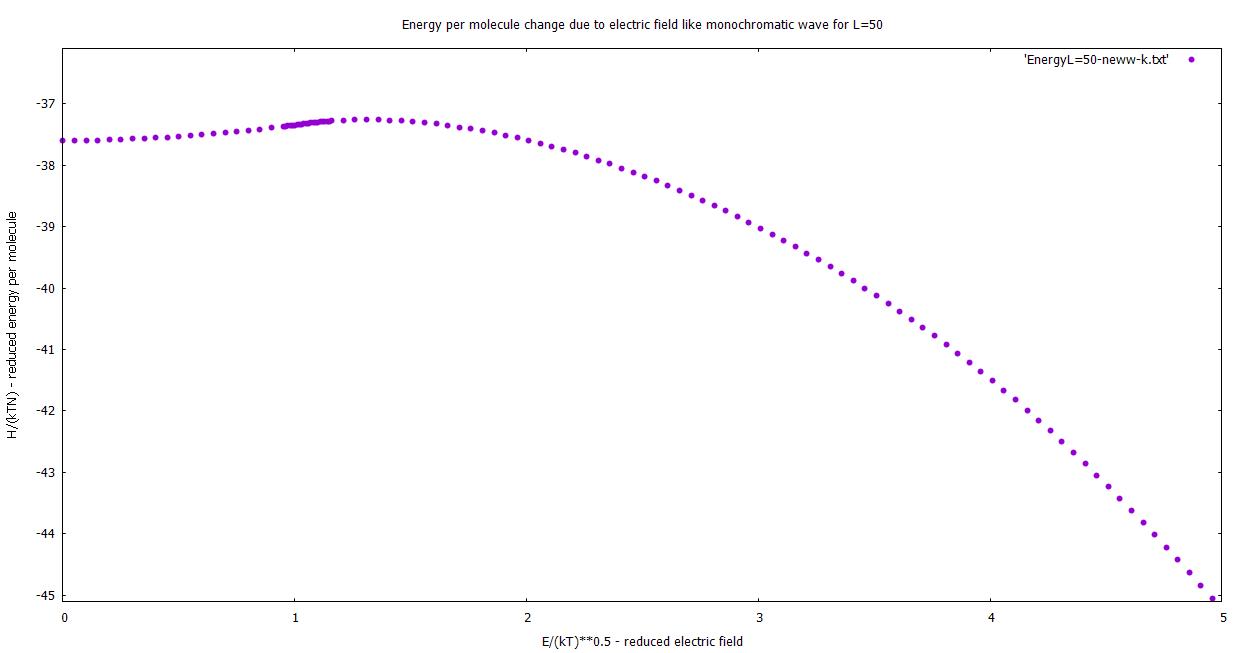
Analogicznie wykonano symulacje dla pola elektrycznego jako fali płaskiej monochromatycznej, którego wykres umieszczono poniżej:



W przypadku pola elektrycznego jako fali płaskiej monochromatycznej z zakresu światła widzialnego pomarańczowego o parametrach:

Próg Freedericksza w tym wypadku wyniósł około:

Wyliczono również energię całego kryształu, wykres znajduje się na kolejnej stronie:



Gdzie korzystano z hamiltonianu (pominięto trzeci człon):

Przy czym wprowadzono wielkości zredukowane: