```
      program tempoMCS
c  Autores: Lucila Marques dos Reis, fez a primeira versão deste programa em 1995, para
c  sua dissertação de mestrado, com um dos objetivos sendo o de especificar as coordenadas
c de todas as 103.346 CSA.
c Mais tarde, em 2000, Roosevelt Alves da Silva introduziu os cálculos referentes ao
c potencial  estéreo-químico, heuristicamente desenvolvido,  utilizando um arquivo de
c entrada externo, composto à mão, durante a sua tese de doutorado.
c Em 2003, Maria Eulália Pinto Tarragó introduziu outros cálculos neste programa para c
sua tese de doutorado, como os de grandezas térmicas.
c Inês Regina Silva, a partir de 2004, produziu uma outra versão do programa onde
automatizou c tanto os arquivos de entrada de dados, quanto o  design  das "proteínas",
durante os  c trabalhos de sua tese de doutorado.
c
c
c Parte-se de uma configuração aleatória sem contatos nativos, e, para cada semente,
c este programa calcula o tempo (MCS) para determinada configuração encontrar a sua
c configuração nativa, dentro da janela de tempo mcs.
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      implicit real*8 (a-h,o-z)
c kcn = contatos nativos
c kcn2 = contatos nativos (ida e volta)
c ns = loop da estratificação
c nd = loop dos descartes
c ne = total de estruturas
      parameter (idim=27,kcn=28,kcn2=28*2,idfmon=11,ns=10,nd=5,ne=51704)
c khetero = número de repulsões
c ntx = número de sementes
c mcs = número de passos MC
      parameter (khetero=44,ntx=15,mcs=100000)
c xlmos = vetor das hidrofobocidades de cada idim monômero
      dimension ix(-2:idim+3),iy(-2:idim+3),iz(-2:idim+3),xlmos(idim)
c ict = vetor das repulsões
      dimension xintera(idim,idim),isemt(ntx),ict(khetero)
c ir = contato com o solvente
c irj = variações dos contatos topológicos
c itl = seqüência de onze letras da estrutura desejada
c ist = seqüência de E, S, T
c mcc = matriz dos contatos topológicos
      dimension ir(idim),irj(idim),itl(idim),ist(idim),mcc(idim,5)
      dimension inactv(kcn),kvntv(kcn2)
      dimension aran(500000),vetsc(idfmon),vetsnm(idfmon)
c ixcd,iycd,izcd = vetores das coordenadas básicas (coord.dat)
      dimension ixcd(idim),iycd(idim),izcd(idim),ilin(idim)
c ixco,iyco,izco=vetores das coordenadas da configuração ie
      dimension ixco(idim),iyco(idim),izco(idim)
c Saída tempo00000.dat = tempo MCS para cada conformação enovelar com potencial
c estéreo-químico para cada ntx
```

```
      character*14 flin
      character*10 idt
      character*4 ext
      common/energya/enere,hidro,temp,id,mon
      common/sequencia/xlmos,xintera,ix,iy,iz
      common/coordenada/inactv,kvntv
      common/coordconfig/ixco,iyco,izco
      common/ordem/inat
      common/sement/isem
      common/seeder/iseed,icont,aran
      common/ranseq/rlist(532),np1,np2
c Arquivo de entrada de dados
c repulsao = repulsões entre os monômeros
c coord = posição (x,y,z) dos idim monômeros
c hidrof = hidrofobicidade do alfabeto
      open(1,file='repulsao.dat',status='old')
      open(3,file='coord.dat',status='old')
      open(4,file='hidrof.dat',status='old')
c posição 6 a 10 em idt = número da estrutura
      idt='tempo00000'
      ext='.dat'
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Lendo as coordenadas básicas para os idim monômeros
      do ico = 1,idim
            read(3,*) ixcd(ico),iycd(ico),izcd(ico)
      end do
      close (3)
c Lendo a hidrofobicidade de cada letra (a,b,c,d,e,f,g,h,i,m,r)
      do ii = 1,idfmon
            read(4,*) vetsnm(ii)
      end do
      close (4)
c khetero = número de repulsões
      do i = 1,khetero
            read(1,*) ict(i)
      end do
      close(1)
c Gerando e guardando as sementes
      call gsnr (ntx,isemt)
c Guardando os idfmon tipos de letras
c a=1, b=2, c=3, d=4 ,e=5, f=6, g=7, h=8, i=9, m=10, r=11
      vetsc(1) = 1
      vetsc(2) = 2
      vetsc(3) = 3
      vetsc(4) = 4
      vetsc(5) = 5
      vetsc(6) = 6
```

```
      vetsc(7) = 7
      vetsc(8) = 8
      vetsc(9) = 9
      vetsc(10) = 10
      vetsc(11) = 11
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Começando o cálculo do tempo MCS para enovelar para cada estrutura CSA
      do 1000 nii = 1,ne
            ie = nii
            if(ie.lt.10) write(idt(10:10),'(i1)')ie
            if(ie.ge.10.and.ie.lt.100) write(idt(9:10),'(i2)')ie
            if(ie.ge.100.and.ie.lt.1000) write(idt(8:10),'(i3)')ie
            if(ie.ge.1000.and.ie.lt.10000) write(idt(7:10),'(i4)')ie
            if(ie.ge.10000.and.ie.lt.100000) write(idt(6:10),'(i5)')ie
            flin = idt//ext
            open(20,file=flin,form='formatted',status='unknown')
c Lendo a seqüência de monômeros da ie configuração desejada
c cfg51704.dat = seqüência posicional dos idim monômeros
            open(2,file='cfg51704.dat',status='old')
            do ibg = 1,ie
                  read(2,*) (ilin(iseq),iseq=1,idim)
c Guardando as coordenadas para cada monômero da ie configuração desejada
                  do ipos = 1,idim
                        imon = ilin(ipos)
                        ixco(ipos)=ixcd(imon)
                        iyco(ipos)=iycd(imon)
                        izco(ipos)=izcd(imon)
                  end do
            end do
            close(2)
c Montando o vetor de contatos nativos da configuração desejada
            call natcont(kcn2,kvntv)
c Compondo o vetor itl da seqüência de letras da configuração
            call tenl(mcc,ir,irj,itl,ist)
c Zerando a matriz de interações para os monômeros
            do i = 1,idim
                  do j = 1,idim
                        xintera(i,j)=0.
                  end do
            end do
c Gerando a matriz de interação entre os monômeros e verificando se a
c seqüência de letras combinadas não e uma repulsão
            call matintera(itl,ict,vetsc,vetsnm)
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
c Iniciando os loops para cada ntx semente isemt
            do 5 ils = 1,ntx
                  iseed = isemt(ils)
```

```
                call ran0
                call ran2(aran,500000)
                icont=1
                hidro = 1.0/0.98
                enere=0.0
                temp=0.8
c Montando o vetor da posição inicial dos monômeros na rede
                ix(1) = 60
                iy(1) = 60
                iz(1) = 60
                call posini
c Valores Fantasmas de monômeros para os movimentos utilizados
                do i = 1,3
                        k = i - 3
                        ix(k) = 100000000
                        iy(k) = 100000000
                        iz(k) = 100000000
                        k = i + idim
                        ix(k) = 100000000
                        iy(k) = 100000000
                        iz(k) = 100000000
                end do
c Inicio do Loop MCS
                do 10 jjj = 1,mcs
                        do 20 jaux= 1,6
c Inicio do Loop Estratificação (ns = 10)
                                do 30 ip=1,ns
c Inicio do Loop Descartes (nd = 5)
                                        do 40 ib=1,nd
c Inicio do Loop que Movimenta os Monômeros (idim=27)
                                                do 50 i=1,idim
c Sorteio do Monômero
                                                        rand=aran(icont)
                                                        icont=icont+1
                                                        if(icont.gt.499990)then
                                                                call ran2(aran,500000)
                                                                icont=1
                                                        end if
                                                mon = int(rand*(idim-0.000001)) + 1
c Inicio dos Movimentos
c (1) Verificando se é movimento Manivela
c Neste caso, o monômero mon e o segundo dos quatro monômeros utilizados
c neste movimento
                                                im = ix(mon + 2) - ix(mon - 1)
                                                jm = iy(mon + 2) - iy(mon - 1)
                                                km = iz(mon + 2) - iz(mon - 1)
                                                nm = im*im + jm*jm + km*km
```

```
                                        if(nm.eq.1) then
                                              id = -1
                                        call crankschaft(im,jm,km)
                                              goto 51
                                        else
c Neste caso, o monômero mon e o terceiro dos quatro monômeros utilizados
c neste movimento
                                  im = ix(mon + 1) - ix(mon -2)
                                  jm = iy(mon + 1) - iy(mon -2)
                                  km = iz(mon + 1) - iz(mon- 2)
                                  nm = im*im + jm*jm + km*km
                                              if(nm.eq.1) then
                                                    id = 1
                                        call crankschaft(im,jm,km)
                                                    goto 51
                                              end if
                                        end if
c (2) Verificando se é movimento de Fim de Cadeia
                                        if(mon.eq.1) then
                            call end1(ix(1),iy(1),iz(1),ix(2),iy(2),iz(2))
                                              goto 51
                                        end if
                                        if(mon.eq.idim) then
      call end1(ix(idim),iy(idim),iz(idim),ix(idim-1),iy(idim-1),iz(idim-1))
                                              goto 51
                                        end if
c (3) Verificando se é movimento de Canto
                                  call corner(ix(mon),iy(mon),iz(mon))
51                                      continue
c Fim do Loop (50) que movimenta os monômeros
                                        call connatv
                                        if(inat.eq.kcn) goto 200
                                        if(jjj.eq.(mcs-1)) goto 200
50                                  continue
c Fim do Loop (40) descartes
40                            continue
c Fim do Loop (30) estratificação
30                          continue
20                    continue
c Fim do Loop (10) MCS
10              continue
200             continue
c Escrevendo no arquivo de saída tempo00000.dat
              write(20,*) ils,jjj
05            continue
      close(unit=20)
      close(unit=30)
```

```
1000    continue
        stop
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine gsnr(ntx,isemt)
c Esta sub-rotina gera o total de ntx sementes
        implicit real*8(a-h,o-z)
        dimension isemt(ntx)
        common/sement/isem

        irn = 1
77      if(irn.le.ntx) then
                call randomic(randu)
                j = int(randu*10**7)
                if (j.gt.1000000.and.mod(j,2).ne.0) then
                        isemt(irn) = j
                        irn = irn + 1
                        goto 77
                end if
                goto 77
        end if
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine natcont(kcn2,kvntv)
c Encontra os contatos nativos da configuração desejada
        implicit real*8(a-h,o-z)
        parameter (idim=27)
        dimension ixco(idim),iyco(idim),izco(idim),kvntv(kcn2)
        common/coordconfig/ixco,iyco,izco

c Calculando a distância entre dois monômeros não adjacentes
        ic = 1
        do i = 1,idim-3
                do j = i+3,idim
                        dx2 = (ixco(i)-ixco(j))**2
                        dy2 = (iyco(i)-iyco(j))**2
                        dz2 = (izco(i)-izco(j))**2
                        d2 = dx2+dy2+dz2
c Se a distância d2 for 1, os monômeros estão em contato topológico
                        if(d2.eq.1.0) then
                                kvntv(ic) = i*100 + j
                                kvntv(ic+1) = j*100 + i
                                ic = ic + 2
                        end if
                end do
        end do
```

```
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine tenl(mcc,ir,irj,itl,ist)
c Calcula a seqüência de onze letras da configuração desejada
      implicit real*8 (a-h, o-z)
      implicit integer (i-n)
      parameter (idim=27)
      dimension mcc(idim,5),ixco(idim),iyco(idim),izco(idim)
      dimension ir(idim),irj(idim),itl(idim),ist(idim)
      common/coordconfig/ixco,iyco,izco

c Montando a matriz de contatos topológicos para cada monômero
      do ij = 1,idim
            do iij = 1,5
                  mcc(ij,iij) = 0
            end do
      end do
      call top(mcc)
c Montando a seqüência E, S, T para a configuração.
      call est(ist)
c Tomando o maior e o menor valor para x, y, z
      ixma = ixco(1)
      iyma = iyco(1)
      izma = izco(1)
      ixmi = ixco(1)
      iymi = iyco(1)
      izmi = izco(1)
      do ird = 2,idim
            if(ixco(ird).gt.ixma) ixma = ixco(ird)
            if(ixco(ird).lt.ixmi) ixmi = ixco(ird)
            if(iyco(ird).gt.iyma) iyma = iyco(ird)
            if(iyco(ird).lt.iymi) iymi = iyco(ird)
            if(izco(ird).gt.izma) izma = izco(ird)
            if(izco(ird).lt.izmi) izmi = izco(ird)
      end do
      do id = 1,idim
            ir(id) = 90
      end do
c As onze letras.
      ka = 1
      kb = 2
      kc = 3
      kd = 4
      ke = 5
      kf = 6
      kg = 7
```

```
      kh = 8
      ki = 9
      km = 10
      kr = 11
c Classificando o contato entre o monômero e o solvente (ir)
c Para o centro do cubo, ir = 0, irj = 1, itl = kr
      imx = (ixma+ixmi)/2
      imy = (iyma+iymi)/2
      imz = (izma+izmi)/2
      do id = 1,idim
            if(imx.eq.ixco(id).and.imy.eq.iyco(id).and.imz.eq.izco(id)) then
                  ir(id)=0
                  irj(id)=1
                  itl(id)=kr
                  icenter = id
            end if
      end do
c Para o vértice do cubo, ir = 3, irj = 1
c Se idim = 1 ou idim = idim, itl = m, em outro caso itl = kc
      do id = 1,idim
            call dist(icenter,id,di,dj,dk)
c A distância entre o centro e o vértice do cubo é (1,1,1)
            if ((di.eq.1.).and.(dj.eq.1.).and.(dk.eq.1.)) then
                  ir(id)=3
                  irj(id)=1
                  if (id.eq.1.or.id.eq.idim) then
                        itl(id)=km
                  else
                        itl(id)=kc
                  end if
            end if
      end do
c Para o centro das faces do cubo, ir = 1. Para valores de itl e irj, o conjunto de ist letras
c será usado.
      do 700 id = 1,idim
            if(id.eq.icenter) goto 700
            call dist(icenter,id,di,dj,dk)
c A distância entre o centro do cubo e o centro das faces é (1,0,0) ou (0,1,0) ou (0,0,1)
            if ((di.gt.1.).or.(dj.gt.1.).or.(dk.gt.1.)) goto 700
c Checando a diagonal
            if (di.eq.dj.and.di.eq.1.) goto 700
            if (di.eq.dk.and.dk.eq.1.) goto 700
            if (dj.eq.dk.and.dj.eq.1.) goto 700
c Acumulando os contados S e T
            iss = 0
            itt = 0
            do ic = 1,5
```

```
                        ja = mcc(id,ic)
c O monômero no centro do cubo não é contado como vizinho topológico
                        if(ja.eq.0) goto 650
                        if(ja.eq.icenter) goto 550
                        if(ist(ja).eq.1) iss = iss+1
                        if(ist(ja).eq.0) itt = itt+1
550                     continue
                end do
650             continue
c Se iss>=itt, irj = 1, outro caso, irj = 2
                if(iss.ge.itt) then
                        ir(id) = 1
                        irj(id) = 1
                        itl(id) = ka
                else
                        ir(id) = 1
                        irj(id) = 2
                        itl(id) = kh
                end if
700     continue
c Para o centro das arestas no cubo, ir = 2
        do id = 1,idim
                if(ir(id).eq.90) then
                        ir(id)=2
c Para valores de irj e itl, é necessário conhecer os vizinhos topológicos
                        i1 = mcc(id,1)
                        i2 = mcc(id,2)
                        if(itl(i1).eq.1.and.itl(i2).eq.1) then
                                irj(id) = 1
                                itl(id) = kb
                        end if
                        if(itl(i1).eq.1.and.itl(i2).eq.8) then
                                irj(id) = 2
                                itl(id) = kg
                        end if
                        if(itl(i2).eq.1.and.itl(i1).eq.8) then
                                irj(id) = 2
                                itl(id) = kg
                        end if
                        if(itl(i1).eq.8.and.itl(i2).eq.8) then
                                irj(id) = 2
                                itl(id) = kg
                        end if
                        if(itl(i1).eq.1.and.itl(i2).eq.10) then
                                irj(id) = 3
                                itl(id) = kf
                        end if
```

```
                    if(itl(i2).eq.10.and.itl(i1).eq.1) then
                        irj(id) = 3
                        itl(id) = kf
                    end if
                    if(itl(i1).eq.1.and.itl(i2).eq.3) then
                        irj(id) = 3
                        itl(id) = kf
                    end if
                    if(itl(i2).eq.1.and.itl(i1).eq.3) then
                        irj(id) = 3
                        itl(id) = kf
                    end if
                    if(itl(i1).eq.3.and.itl(i2).eq.8) then
                        irj(id) = 4
                        itl(id) = ki
                    end if
                    if(itl(i2).eq.3.and.itl(i1).eq.8) then
                        irj(id) = 4
                        itl(id) = ki
                    end if
                    if(itl(i1).eq.8.and.itl(i2).eq.10) then
                        irj(id) = 5
                        itl(id) = ke
                    end if
                    if(itl(i2).eq.8.and.itl(i1).eq.10) then
                        irj(id) = 5
                        itl(id) = ke
                    end if
                    if(itl(i1).eq.3.and.itl(i2).eq.3) then
                        irj(id) = 6
                        itl(id) = kd
                    end if
                    if(itl(i1).eq.3.and.itl(i2).eq.10) then
                        irj(id) = 6
                        itl(id) = kd
                    end if
                    if(itl(i1).eq.10.and.itl(i2).eq.3) then
                        irj(id) = 6
                        itl(id) = kd
                    end if
                    if(itl(i1).eq.10.and.itl(i2).eq.10) then
                        irj(id) = 6
                        itl(id) = kd
                    end if
            end if
end do
return
```

```
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine top(mcc)
c Compõe a matriz de contatos topológicos
       implicit real*8 (a-h, o-z)
      implicit integer (i-n)
      parameter (idim=27)
      dimension mcc(idim,5),ixco(idim),iyco(idim),izco(idim)
      common/coordconfig/ixco,iyco,izco

      do iat = 1,idim
            iac = 1
            do ji = 1,idim
                  if(ji.eq.iat) goto 100
                  if(ji.eq.iat-1.or.ji.eq.iat+1) goto 100
                  if(ji.eq.iat-2.or.ji.eq.iat+2) goto 100
c Checking the distance between monomers to be a topological contact
                  di = abs(ixco(ji)-ixco(iat))
                  dj = abs(iyco(ji)-iyco(iat))
                  dk = abs(izco(ji)-izco(iat))
                  if ((di.gt.1.).or.(dj.gt.1.).or.(dk.gt.1.)) goto 100
c Checking the diagonal
            if (di.eq.dj.and.di.eq.1.) goto 100
                  if (di.eq.dk.and.dk.eq.1.) goto 100
                  if (dj.eq.dk.and.dj.eq.1.) goto 100
c They are topological, the position is separated
                  mcc(iat,iac) = ji
                  iac = iac + 1
100               continue
            end do
      end do
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine est(ist)
c Verifies if the monomer is in position straight or turn between
c its neighbors
c 2 = end of sequence
c 1 = straight
c 0 = turn
     implicit real*8 (a-h, o-z)
     implicit integer (i-n)
      parameter (idim=27)
      dimension ixco(idim),iyco(idim),izco(idim),ist(idim)
      common/coordconfig/ixco,iyco,izco

      ist(1) = 2
```

```
      ist(idim) = 2
c Taking the vectors for to label the monomers
      do id = 2,idim-1
            i1 = ixco(id-1)
            i2 = ixco(id)
            i3 = ixco(id+1)
            j1 = iyco(id-1)
            j2 = iyco(id)
            j3 = iyco(id+1)
            l1 = izco(id-1)
            l2 = izco(id)
            l3 = izco(id+1)
            iv1x = i2-i1
            iv1y = j2-j1
            iv1z = l2-l1
            iv2x = i3-i2
            iv2y = j3-j2
            iv2z = l3-l2
c Doing the scalar product. If this result is iqual 1, the monomer id
c is s, another case, is t
            ipe = iv1x*iv2x + iv1y*iv2y + iv1z*iv2z
            if(ipe.eq.1) ist(id) = 1
            if(ipe.ne.1) ist(id) = 0
      end do
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine dist(it,id,di,dj,dk)
c Calculates the distance between id and icenter
     implicit real*8 (a-h, o-z)
     implicit integer (i-n)
      parameter (idim=27)
      dimension ixco(idim),iyco(idim),izco(idim)
      common/coordconfig/ixco,iyco,izco

c Checking the distance between monomers to be a topological contact
      di = abs(ixco(id)-ixco(it))
      dj = abs(iyco(id)-iyco(it))
      dk = abs(izco(id)-izco(it))
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine end1(ix1,iy1,iz1,ix2,iy2,iz2)
c Movimenta os monomeros da ponta da cadeia
      implicit real*8(a-h,o-z)
      parameter(idim=27)
      dimension ix(-2:idim+3),iy(-2:idim+3),iz(-2:idim+3),xlmos(idim)
```

```
      dimension xintera(idim,idim),aran(500000)
      common/sequencia/xlmos,xintera,ix,iy,iz
      common/seeder/iseed,icont,aran
      common/ranseq/rlist(532),np1,np2
      common/energya/enere,hidro,temp,id,mon

      idx = 0
      idy = 0
      idz = 0
c Sorteando a direção do movimento
      rand=aran(icont)
      icont=icont+1
      if(rand.lt.0.16666) then
            idx = 1
      else
            if(rand.lt.0.3333) then
                  idx = -1
            else
                  if(rand.lt.0.5) then
                        idy = 1
                  else
                        if(rand.lt.0.6666) then
                              idy = -1
                        else
                              if(rand.lt.0.8333) then
                                    idz = 1
                              else
                                    idz = -1
                              end if
                        end if
                  end if
            end if
      end if
c Mudando a posição do monômero para a nova direção
      ixend = ix2 + idx
      iyend = iy2 + idy
      izend = iz2 + idz
c Verificando se a posição mudou
      if(ixend.ne.ix1) goto 1
      if(iyend.ne.iy1) goto 1
      if(izend.ne.iz1) goto 1
      goto 2
1     continue
c Verificando se não houve sobreposição de algum monômero
      do 10 is=1,idim
            if(ixend.ne.ix(is)) goto 10
            if(iyend.ne.iy(is)) goto 10
```

```
                if(izend.ne.iz(is)) goto 10
                goto 2
10      continue
c Cálculo da energia
        call energia(ixend,iyend,izend)
2       continue
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine energia(kx2,ky2,kz2)
c Calcula a energia do movimento dos monômeros do final da cadeia e canto
        implicit real*8(a-h,o-z)
        parameter(idim=27)
        dimension ix(-2:idim+3),iy(-2:idim+3),iz(-2:idim+3),xlmos(idim)
        dimension xintera(idim,idim)
        dimension aran(500000)
        common/energya/enere,hidro,temp,id,mon
        common/sequencia/xlmos,xintera,ix,iy,iz
        common/seeder/iseed,icont,aran
        common/ranseq/rlist(532),np1,np2

c Cálculo da energia local do sistema
        call elocal(kx2,ky2,kz2,xivarloc,xivarloc2)
c Verificando a aceitação ou não da nova posição, através do valor da energia
        entrop = -hidro*xivarloc - xivarloc2/temp
        if(entrop.ge.0) goto 11
        rand=aran(icont)
        icont=icont + 1
        prob = exp(entrop)
        if(rand.le.prob) goto 11
        goto 12
11      continue
c Atualiza a energia
        enere = enere + xivarloc
c Atualiza a posição do monômero
        ix(mon) = kx2
        iy(mon) = ky2
        iz(mon) = kz2
12      continue
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine elocal(iloc,jloc,kloc,xivarloc,xivarloc2)
c Calcula a energia local do sistema para os movimentos "end" e "corner"
        implicit real*8(a-h,o-z)
        parameter(idim=27)
        dimension ix(-2:idim+3),iy(-2:idim+3),iz(-2:idim+3),xlmos(idim)
```

```
      dimension xintera(idim,idim)
      common/energya/enere,hidro,temp,id,mon
      common/sequencia/xlmos,xintera,ix,iy,iz

      xidel1=0.
      xidel2=0.
      xivarloc=0.
      xidel11=0.
      xidel22=0.
      do 10 n=1,idim
            xint1x=0.
            xint2x=0.
            xint11x=0.
            xint22x=0.
c Verificando quais monômeros são vizinhos antes do movimento
            il1 = ix(mon) - ix(n)
            jl1 = iy(mon) - iy(n)
            kl1 = iz(mon) - iz(n)
            norm1 = il1*il1 + jl1*jl1 + kl1*kl1
            icova = abs(mon - n)
            if((norm1.eq.1).and.(icova.ne.1))then
                  xint1x = xlmos(mon)+xlmos(n)
                  xint11x= xintera(mon,n)
            end if
            xidel1 = xidel1 + xint1x
            xidel11 = xidel11 + xint11x
c Verificando quais monômeros são vizinhos depois do movimento
            il2 = iloc - ix(n)
            jl2 = jloc - iy(n)
            kl2 = kloc - iz(n)
            norm2 = il2*il2 + jl2*jl2 + kl2*kl2
            if((norm2.eq.1).and.(icova.ne.1))then
                  xint2x = xlmos(mon)+xlmos(n)
                  xint22x = xintera(mon,n)
            end if
            xidel2 = xidel2 + xint2x
            xidel22 = xidel22 + xint22x
10    continue
c Calculando a variação da energia na cadeia
      xivarloc = xidel2 - xidel1
      xivarloc2 = xidel22 - xidel11
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine corner(kcornx,kcorny,kcornz)
c Movimenta os cantos da cadeia
      implicit real*8(a-h,o-z)
```

```
      parameter(idim=27)
      dimension ix(-2:idim+3),iy(-2:idim+3),iz(-2:idim+3),xlmos(idim)
      dimension xintera(idim,idim)
      common/energya/enere,hidro,temp,id,mon
      common/sequencia/xlmos,xintera,ix,iy,iz

c Vetor da nova posição do monômero de canto
      ixcant= ix(mon-1) + ix(mon+1) - kcornx
      iycant= iy(mon-1) + iy(mon+1) - kcorny
      izcant= iz(mon-1) + iz(mon+1) - kcornz
c Verifica se mudou de posição
      if(kcornx.ne.ixcant) goto 2
      if(kcorny.ne.iycant) goto 2
      if(kcornz.ne.izcant) goto 2
      goto 3
2     continue
c Verifica se não houve sobreposição de monômeros
      do 4 is=1,idim
            if(ixcant.ne.ix(is)) goto 4
            if(iycant.ne.iy(is)) goto 4
            if(izcant.ne.iz(is)) goto 4
            goto 3
4     continue
c Calculando a energia do movimento de Canto
      call energia(ixcant,iycant,izcant)
3     continue
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine crankschaft(im1,jm1,km1)
c Movimenta os monomeros que estao como manivela
      implicit real*8(a-h,o-z)
      parameter(idim=27)
      dimension ix(-2:idim+3),iy(-2:idim+3),iz(-2:idim+3),xlmos(idim)
      dimension xintera(idim,idim),aran(500000)
      common/energya/enere,hidro,temp,id,mon
      common/sequencia/xlmos,xintera,ix,iy,iz
      common/seeder/iseed,icont,aran
      common/ranseq/rlist(532),np1,np2

      imon = mon + id
c Determinando o vetor distância
      jdx = ix(mon) - ix(imon)
      jdy = iy(mon) - iy(imon)
      jdz = iz(mon) - iz(imon)
      rand=aran(icont)
      icont=icont + 1
```

```
c Calculando o produto vetorial entre m1 e distancia
      if(rand.lt.0.25)then
            ivet = jdy*km1 - jdz*jm1
            jvet = jdz*im1 - jdx*km1
            kvet = jdx*jm1 - jdy*im1
      else
            if(rand.lt.0.5) then
                  ivet = jm1*jdz - km1*jdy
                  jvet = km1*jdx - im1*jdz
                  kvet = im1*jdy - jm1*jdx
            else
                  if(rand.lt.0.75) then
                        ivet1 = jm1*jdz - km1*jdy
                        jvet1 = km1*jdx - im1*jdz
                        kvet1 = im1*jdy - jm1*jdx
                        ivet = jvet1*km1 - kvet1*jm1
                        jvet = kvet1*im1 - ivet1*km1
                        kvet = ivet1*jm1 - jvet1*im1
                  else
                        ivet2 = jdy*km1 - jdz*jm1
                        jvet2 = jdz*im1 - jdx*km1
                        kvet2 = jdx*jm1 - jdy*im1
                        ivet = jvet2*km1 - kvet2*jm1
                        jvet = kvet2*im1 - ivet2*km1
                        kvet = ivet2*jm1 - jvet2*im1
                  end if
            end if
      end if
c Posição dos monômeros da manivela apos o movimento
      if(id.eq.-1) kmon = mon - 1
      if(id.eq.1) kmon = mon - 2
      ix2 = ix(kmon) + ivet
      iy2 = iy(kmon) + jvet
      iz2 = iz(kmon) + kvet
      ix3 = ix2 + im1
      iy3 = iy2 + jm1
      iz3 = iz2 + km1
c Verifica se não houve sobreposição de monômeros
      ksobre=0
      call overlap(ix2,iy2,iz2,ix3,iy3,iz3,ksobre)
      if(ksobre.eq.0) then
c Calcula a nova energia devido o movimento
            call eneman(ix2,iy2,iz2,ix3,iy3,iz3)
      end if
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
```

```
      subroutine eneman(mx2,my2,mz2,mx3,my3,mz3)
c Calcula a variação da energia devido o movimento da manivela
      implicit real*8(a-h,o-z)
      parameter(idim=27)
      dimension ix(-2:idim+3),iy(-2:idim+3),iz(-2:idim+3),xlmos(idim)
      dimension xintera(idim,idim),aran(500000)
      common/energya/enere,hidro,temp,id,mon
      common/sequencia/xlmos,xintera,ix,iy,iz
      common/seeder/iseed,icont,aran
      common/ranseq/rlist(532),np1,np2

      xivarene = 0.
      xidel1a = 0.
      xidel1aa= 0.
      xidel2a = 0.
      xidel2aa = 0.
      xidel1b = 0.
      xidel1bb = 0.
      xidel2b = 0.
      xidel2bb = 0.
      if(id.eq.-1) then
            iaa = mon
            iab = mon + 1
      else
            iaa = mon - 1
            iab = mon
      end if
      do 10 n=1,idim
            xint1am=0.
            xint1aam=0.
            xint2am=0.
            xint2aam=0.
            xint1bm=0.
            xint1bbm=0.
            xint2bm=0.
            xint2bbm=0.
c Verificando quais monômeros são vizinhos antes do movimento
            imon1a = ix(iaa) - ix(n)
            jmon1a = iy(iaa) - iy(n)
            kmon1a = iz(iaa) - iz(n)
            imon2a = ix(iab) - ix(n)
            jmon2a = iy(iab) - iy(n)
            kmon2a = iz(iab) - iz(n)
c Verificando quais monômeros são vizinhos depois do movimento
            lx1 = ix(iaa)
            ly1 = iy(iaa)
            lz1 = iz(iaa)
```

```
                lx2 = ix(iab)
                ly2 = iy(iab)
                lz2 = iz(iab)
                ix(iaa)=mx2
                iy(iaa)=my2
                iz(iaa)=mz2
                ix(iab)=mx3
                iy(iab)=my3
                iz(iab)=mz3
                imon1b = mx2 - ix(n)
                jmon1b = my2 - iy(n)
                kmon1b = mz2 - iz(n)
                imon2b = mx3 - ix(n)
                jmon2b = my3 - iy(n)
                kmon2b = mz3 - iz(n)
c Calculando o produto vetorial antes e depois do movimento
                norm1a =imon1a*imon1a+jmon1a*jmon1a+kmon1a*kmon1a
                norm2a =imon2a*imon2a+jmon2a*jmon2a+kmon2a*kmon2a
                norm1b =imon1b*imon1b+jmon1b*jmon1b+kmon1b*kmon1b
                norm2b =imon2b*imon2b+jmon2b*jmon2b+kmon2b*kmon2b
                ix(iaa)=lx1
                iy(iaa)=ly1
                iz(iaa)=lz1
                ix(iab)=lx2
                iy(iab)=ly2
                iz(iab)=lz2
                icva = abs(iaa - n)
                icvb = abs(iab - n)
                if((norm1a.eq.1).and.(icva.ne.1)) then
                        xint1am = xlmos(iaa)+xlmos(n)
                        xint1aam = xintera(iaa,n)
                end if
                if((norm2a.eq.1).and.(icvb.ne.1))then
                        xint2am = xlmos(iab) + xlmos(n)
                        xint2aam = xintera(iab,n)
                end if
                if((norm1b.eq.1).and.(icva.ne.1))then
                        xint1bm = xlmos(iaa) + xlmos(n)
                        xint1bbm = xintera(iaa,n)
                end if
                if((norm2b.eq.1).and.(icvb.ne.1))then
                        xint2bm = xlmos(iab) + xlmos(n)
                        xint2bbm = xintera(iab,n)
                end if
                xidel1a = xidel1a + xint1am
                xidel1aa = xidel1aa + xint1aam
                xidel2a = xidel2a + xint2am
```

```
                xidel2aa = xidel2aa + xint2aam
                xidel1b = xidel1b + xint1bm
                xidel1bb = xidel1bb + xint1bbm
                xidel2b = xidel2b + xint2bm
                xidel2bb = xidel2bb + xint2bbm
10      continue
c Variação da energia devido o movimento manivela
        xivarene = (xidel1b - xidel1a) + (xidel2b - xidel2a)
        xivarene2 = (xidel1bb - xidel1aa) + (xidel2bb - xidel2aa)
c Verifica a aceitação ou não da variação da energia
        entrop = -hidro*xivarene - xivarene2/temp
        if(entrop.ge.0) goto 11
        rand=aran(icont)
        icont=icont+1
        prob = exp(entrop)
        if(rand.le.prob) goto 11
        goto 12
11      continue
c Atualizando a energia
        enere = enere + xivarene
c Atualizando as posições dos monômeros da manivela
        ix(iaa) = mx2
        iy(iaa) = my2
        iz(iaa) = mz2
        ix(iab) = mx3
        iy(iab) = my3
        iz(iab) = mz3
12      continue
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine overlap(iov1,jov1,kov1,iov2,jov2,kov2,ksobre)
c Verifica se houve sobreposição dos monômeros apos o movimento da manivela
        implicit real*8(a-h,o-z)
        parameter(idim=27)
        dimension ix(-2:idim+3),iy(-2:idim+3),iz(-2:idim+3),xlmos(idim)
        dimension xintera(idim,idim)
        common/sequencia/xlmos,xintera,ix,iy,iz

        ksobre = 0
        do 10 i=1,idim
                if(iov1.ne.ix(i)) goto 15
        if(jov1.ne.iy(i)) goto 15
        if(kov1.ne.iz(i)) goto 15
        ksobre = 1
        goto 20
15      continue
```

```
      if(iov2.ne.ix(i)) goto 10
      if(jov2.ne.iy(i)) goto 10
      if(kov2.ne.iz(i)) goto 10
      ksobre = 1
      goto 20
10    continue
20    continue
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine posini
c Posiciona uma seqüência de n monômeros aleatoriamente sem nenhum contato
      implicit real*8 (a-h,o-z)
      parameter(idim=27)
      dimension ix(-2:idim+3),iy(-2:idim+3),iz(-2:idim+3),xlmos(idim)
      dimension xintera(idim,idim),aran(500000)
      common/sequencia/xlmos,xintera,ix,iy,iz
      common/seeder/iseed,icont,aran
      common/ranseq/rlist(532),np1,np2

      do 50 i=2,idim
2           continue
      idx= 0
      idy = 0
      idz = 0
      rand=aran(icont)
      icont=icont+1
      if(rand.lt.0.16666) then
            idx = 1
      else
            if(rand.lt.0.3333) then
                  idx = -1
            else
                  if(rand.lt.0.5) then
                        idy = 1
                  else
                        if(rand.lt.0.6666) then
                              idy = -1
                        else
                              if(rand.lt.0.8333) then
                                    idz = 1
                              else
                                    idz = -1
                              end if
                        end if
                  end if
            end if
```

```
      end if
c Mudando a posição do monômero para a nova direção
      ixend = ix(i-1) + idx
      iyend = iy(i-1) + idy
      izend = iz(i-1) + idz
c Verificando se não houve sobreposição e se não são vizinhos topológicos
      do 10 ls=1,i-1
            itin = ixend - ix(ls)
            jtin = iyend - iy(ls)
            ktin = izend - iz(ls)
            nortin = itin*itin + jtin*jtin + ktin*ktin
            icva=abs(i-ls)
            if((nortin.ne.1).or.(icva.eq.1)) then
                  if(ixend.ne.ix(ls)) goto 10
                  if(iyend.ne.iy(ls)) goto 10
                  if(izend.ne.iz(ls)) goto 10
                  goto 2
            end if
            goto 2
10    continue
      ix(i)=ixend
      iy(i)=iyend
      iz(i)=izend
50    continue
      return
      end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine connatv
c Calcula o número de contatos nativos de uma dada configuração
      implicit real*8(a-h,o-z)
      parameter(idim=27,kcn=28,kcn2=28*2)
      dimension ix(-2:idim+3),iy(-2:idim+3),iz(-2:idim+3),xlmos(idim)
      dimension xintera(idim,idim),inactv(kcn),kvntv(kcn2)
      common/sequencia/xlmos,xintera,ix,iy,iz
      common/coordenada/inactv,kvntv
      common/ordem/inat

      do it = 1,kcn
            inactv(it) = 0
      end do
      inat = 0
      ict28=0
      do i=1,idim-3
            do j=i+3,idim
                  iti = ix(i) - ix(j)
                  itj = iy(i) - iy(j)
                  itk = iz(i) - iz(j)
```

```
c Produto vetorial
                        normt = iti*iti + itj*itj + itk*itk
                        icovt = abs(i - j)
c Se produto vetorial.eq.1 e forem vizinhos topologicos, e contato nativo
                        if((normt.eq.1).and.(icovt.ne.1)) then
                                ict28 = ict28 + 1
                                inactv(ict28) = i*100 + j
                        end if
                end do
        end do
        do m=1,kcn2
                do mn = 1,kcn
                        if(kvntv(m).eq.inactv(mn)) then
                                inat=inat+1
                        end if
                end do
        end do
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine matintera(itl,ict,vetsc,vetsnm)
c Subrotina que gera a matriz de interação entre os monômeros do tipo
c repulsão e especifica
        implicit real*8 (a-h,o-z)
        parameter (idim=27,khetero=44,ntx=15,idfmon=11)
        dimension xintera(idim,idim)
        dimension vetsnm(idfmon),ix(-2:idim+3),iy(-2:idim+3),iz(-2:idim+3)
        dimension xlmos(idim),vetsc(idfmon),itl(idim),ict(khetero)
        common/sequencia/xlmos,xintera,ix,iy,iz

c Relacionando cada monômero com sua letra
        do 66 mn=1,idim
                do idf=1,idfmon
                        if(itl(mn).eq.vetsc(idf)) then
                                xlmos(mn)=vetsnm(idf)
                                goto 66
                        end if
                end do
66      continue
c Combinando a seqüência de letras dos monômeros
        do i=1,idim
                do j=1,idim
                        ic1 = itl(i)*100+itl(j)
                        ic2 = itl(j)*100+itl(i)
c Verificando se a combinação de letras dos monômeros não é uma repulsão
                        do k=1,khetero
                                if((ic1.eq.ict(k)).or.(ic2.eq.ict(k)))then
```

```
                                        xintera(i,j) = 20
                                        xintera(j,i) = 20
                                end if
                        end do
                end do
        end do
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine ran0
        implicit real*8(a-h,o-z)
        dimension aran(500000)
        integer*4 max,np1,np2,np3
        integer*4 table(128),mults(5)
        common/ranh/amaxin,max,n,mult,table,mults,n2
        common/ranseq/rlist(532),np1,np2
        common/seeder/iseed,icont,aran

        mults(1)=189
        mults(2)=187
        mults(3)=195
        mults(4)=181
        mults(5)=197
        next=iseed
        lext=next/32766
        next =next - lext*32766 + 1
        n = iabs(next/2)*2 +1
        max=32768
        amaxin=1.0/max
        mult=197
        do 15 j=1,128
                n=n*mult
                l=n/32768
                n=n-l*32768
                table(j)=n
15      continue
c INITIALIZE RAN 532
        np1 = 532
        np2 = 37
        fact = 1.0/512.0
        fact2=fact*fact
        np3=3*532
        call ran1(aran,np3)
        do 50 k=1,532
                x=aran(k)-aran(k+532)*fact-aran(k+2*532)*fact2
                if(x.le.0.0) x=x + 1.0
                rlist(k)=x
```

```
50      continue
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine ran1(aran,nran)
        implicit real*8(a-h,o-z)
        dimension aran(500000)
        integer*4 max,nran,k
        integer*4 table(128),mults(5)
        common/ranh/amaxin,max,n,mult,table,mults,n2

c nran não pode ser maior do que 500000
        do 1 k=1,nran
                kmult=k-(k/5)*5 +1
                mult = mults(kmult)
                n=n*mult
                l=n/32768
                n=n-l*32768
                m=n/256 + 1
                n2 = table(m)
                aran(k) = n2*amaxin
                n2 = n2*mult
                l2=n2/32768
                n2 = n2 - l2*32768
                table(m)=n2
1       continue
        return
        end
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
        subroutine ran2(aran,nran)
        implicit real*8 (a-h,o-z)
        dimension aran(500000)
        common/ranseq/rlist(532),np1,np2

c nran não pode ser maior que 500000
        do 1 k=1,nran
                np1=np1 - 1
                if(np1.le.0)np1=532
                np2=np2-1
                if(np2.le.0)np2=532
                x=rlist(np1) -rlist(np2)
                if(x.lt.0.0) x=x+1.0
                rlist(np1)=x
                aran(k)=x
1       continue
        return
        end
```

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccc
      subroutine randomic(randu)
c Pseudo random number generator in a uniform distribution
      common/sement/isem
      real*8 randu

      isem = isem*1103515245  + 453816693
      if (isem .EQ. -2147483648) then
            randu = 0.9999999996
            return
      end if
      randu = dble(isem)/(-2147483648.0D0)
      randu = ABS(randu)
      return
      end
```