# Viðauki

## parameter.f90

```
module parameter
implicit none

!*****************************************************************
!Stærðir kerfisins skilgreindar
!Ath, hér er orkan I skilgreind sem V
!*****************************************************************


integer,parameter::Nx=10,Ny=10,lengd=100000,T_lengd=30
real (kind=8)::V=1.0d0,B=0.0d0,beta=1.0d0,T_max=5.0d0

!beta=1/(k_B*T)

!*****************************************************************
!Hitastigið látið hlaupa frá T=0 í T=T_max í T_lengd skrefum
!*****************************************************************

contains

  subroutine hiti(T,i)

    integer::i
    real (kind=8)::T,h

    h=T_max/T_lengd

    T=i*h

    beta=1/T

  end subroutine hiti


end module parameter
```

## ising.f90

```
program ising
use parameter
implicit none

real (kind=8)::E_avg,G_avg,E_var,G_var,C,X,T
real (kind=8)::E_favg,G_favg,X_favg,C_favg,andrand

integer::i,k,m,n,l
integer,parameter::rand=1

integer      ,dimension(Nx,Ny) ::S
real (kind=8),dimension(rand)  ::E,G,X_vig,C_vig
real (kind=8),dimension(Nx/2-1)::fall_vig,fall_avg


fall_avg=0

open (unit=11,file=’E_avg.txt’,status=’new’)
open (unit=12,file=’G_avg.txt’,status=’new’)
open (unit=13,file=’C.txt’,status=’new’)
open (unit=14,file=’X.txt’,status=’new’)
open (unit=15,file=’Spuni+1.txt’,status=’new’)
open (unit=16,file=’Spuni0.txt’,status=’new’)
open (unit=17,file=’Spuni-1.txt’,status=’new’)
open (unit=18,file=’Fylgni.txt’,status=’new’)

andrand=1/float(rand)

do i=1,T_lengd

   call hiti(T,i)

!*****************************************************************
!Forritið er látið hlaupa oft í gegnum Monte Carlo aðferðina til að
!koma í veg fyrir flökt. Að lokum er meðaltal tekið.
!*****************************************************************

   do k=1,rand

      call undir_ising(S,E_avg,G_avg,E_var,G_var,C,X,fall_vig)
      E(k)=E_avg
      G(k)=G_avg
      C_vig(k)=C
      X_vig(k)=X

      fall_avg = fall_avg + andrand * fall_vig

   end do

   call medaltal(E,E_favg,rand)
   call medaltal(G,G_favg,rand)
   call medaltal(C_vig,C_favg,rand)
   call medaltal(X_vig,X_favg,rand)

   write (11,*),T,E_favg
   write (12,*),T,G_favg
   write (13,*),T,C_favg
   write (14,*),T,X_favg

end do

do m=1,Nx

   do n=1,Ny

      if (S(m,n)==1) then

         write (15,*),m,n

      else if (S(m,n)==0) then

        write (16,*),m,n

      else if (S(m,n)==-1) then

         write (17,*),m,n

      end if

   end do

end do

do l=1,Nx/2-1
```

```
   write (18,*),l,fall_avg(l)

end do

close (11)
close (12)
close (13)
close (14)
close (15)
close (16)
close (17)
close (18)

end program ising
```

### undir_ising.f90

```
subroutine undir_ising(S,E_avg,G_avg,E_var,G_var,C,X,sum_vig)
use parameter
implicit none

real (kind=8)::G,E,z,r,E_flipp,E_avg,G_avg
real (kind=8)::E_var,G_var,C,X,fall,andlengd
integer      ::m,n,i,k,s_old,s_new

real (kind=8),dimension(lengd)    ::E_vig,G_vig
real (kind=8),dimension(Nx/2-1)   ::fall_vig,sum_vig
integer      ,dimension(Nx,Ny)    ::S
integer      ,dimension(Nx+2,Ny+2)::A

sum_vig=0

andlengd=1/float(lengd)

!call fylki(S)          !Ef byrjunarfylkið S er tilviljunarkennt

S=1                !Ef byrjunarfylkið S hefur alla spuna 1
```

```
call afylki(S,A)

call random_seed


!*******************************************************************
!Monte Carlo aðferðin
!*******************************************************************


do i=1,lengd

   call random_number(z)

   call flipp(S,s_old,s_new,m,n,r)

   call hlutfall(S,A,E_flipp,s_new,m,n,r)


   do k=1,Nx/2-1

      call fylgni(S,k,fall)

      fall_vig(k) = fall

   end do


   sum_vig = sum_vig + andlengd * fall_vig


   if (E_flipp<=0) then

      S(m,n)=s_new

      call afylki(S,A)

   else if (z<=r) then

      S(m,n)=s_new

      call afylki(S,A)

   end if
```

```
    call hamiltonian(S,A,E)

    E_vig(i)=E

    call magni(S,G)

    G_vig(i)=G

end do

call medaltal(E_vig,E_avg,lengd)
call medaltal(G_vig,G_avg,lengd)

call stadalfravik(E_vig,E_var)
call stadalfravik(G_vig,G_var)


C=E_var*beta**2

X=G_var*beta


!******************************************************************
!Orkunni E_avg ásamt eðlisvarma C er deilt með stærð kerfisins
!******************************************************************

E_avg=E_avg/float(Nx*Ny)

C=C/float(Nx*Ny)


end subroutine undir_ising
```

### fylki.f90

```
subroutine fylki(S)
```

```
use parameter

real (kind=8),dimension(Nx,Ny)::D
integer       ,dimension(Nx,Ny)::S

!*****************************************************************
!Fylki S búið til með tilviljunarkenndri uppröðun spuna
!*****************************************************************

call random_seed

call random_number(D)

S=D*3
S=S-1

end subroutine fylki
```

---

## afylki.f90

---

```
subroutine afylki(S,A)
use parameter
implicit none

integer,dimension(Nx,Ny)    ::S
integer,dimension(Nx+2,Ny+2)::A

integer::j,k

!*****************************************************************
!Fylki A búið til út frá fylki S vegna lotubundinna jaðarskilyrða
!*****************************************************************

A=0

A(2:Nx+1,2:Ny+1) = S
A(2:Nx+1,1)      = S(1:Nx,Ny)
A(2:Nx+1,Ny+2)   = S(1:Nx,1)
A(1,2:Ny+1)      = S(Nx,1:Ny)
```

```
A(Nx+2,2:Ny+1)    = S(1,1:Ny)


end subroutine afylki
```

## flipp.f90

```
subroutine flipp(S,s_old,s_new,m,n,r)
use parameter
implicit none

real (kind=8)::r,i_rand,j_rand,a_rand

integer,dimension(Nx,Ny)::S

integer::i,j,m,n,s_old,s_new


!*****************************************************************
!Hér er valið sæti (m,n) í spunafylkinu S sem á að snúa, "flippa"
!*****************************************************************

call random_number(i_rand)

call random_number(j_rand)

i=i_rand*Nx

m=i+1

j=j_rand*Ny

n=j+1

s_old=S(m,n)


!*****************************************************************
!Hér er spunaflutningur leyfður frá S=1 í S=0, S=-1 í S=0 og S=0 í
```

```
!S=-1,1. Einnig er hægt að leyfa S=1 í S=-1 og öfugt en það er
!"commentað" út hér
!******************************************************************

call random_number(a_rand)

if (S(m,n)==1) then

!   if (a_rand<0.5d0) then

      s_new=0

!   else

!      s_new=-1

!   end if


else if (S(m,n)==-1) then

!   if (a_rand<0.5d0) then

!      s_new=1

!   else

      s_new=0

!   end if


else if (S(m,n)==0) then

   if (a_rand<0.5d0) then

      s_new=-1

   else

      s_new=1

   end if
end if


end subroutine flipp
```

---

## hlutfall.f90

---

```
subroutine hlutfall(S,A,E_flipp,s_new,m,n,r)
use parameter
implicit none

real (kind=8)::r,E_flipp,E_0,E_t

integer::s_new,m,n

integer,dimension(Nx,Ny)    ::S,S_t
integer,dimension(Nx+2,Ny+2)::A,A_t

!******************************************************************
!Hér er r=exp(-beta*E_flipp) reiknað. Hér stendur X_t fyrir stærðir
!þar sem spuna hefur verið snúið en X fyrir óbreyttan spuna
!******************************************************************


E_0=0d0

E_t=0d0

S_t=S

S_t(m,n)=s_new


call afylki(S_t,A_t)

call Hamiltonian(S,A,E_0)

call Hamiltonian(S_t,A_t,E_t)
```

```
E_flipp=E_t-E_0


r=exp(-beta*E_flipp)


end subroutine hlutfall
```

## hamiltonian.f90

```
subroutine hamiltonian(S,A,E)
use parameter
implicit none

real (kind=8)::sum1,sum2,E

integer,dimension(Nx,Ny)    ::S
integer,dimension(Nx+2,Ny+2)::A

integer::i,j

E=0
sum1=0
sum2=0


do i=2,Nx+1

   do j=2,Ny+1

      sum1=sum1 + A(i,j)*(A(i-1,j) + A(i+1,j) + A(i,j-1) + A(i,j+1))

   end do

end do
```

```
sum2=sum(float(S))

E=-0.5*V*sum1-B*sum2


end subroutine hamiltonian
```

## magni.f90

```
subroutine magni(S,G)
use parameter
implicit none

!*******************************************************************
!Ath, hér er seGlunin M skilgreind sem G
!*******************************************************************

real (kind=8)::G,summa

integer,dimension(Nx,Ny)::S

integer::i,j


summa=0.d0


if (V>0) then

   G=SUM(float(S))/float(SIZE(S))

else if (V<0) then

   do i=1,Nx

      do j=1,Ny

         summa=summa + (-1)**(i+j)*S(i,j)
         end do

      end do

      G=summa

   end if


   end subroutine magni
```

### fylgni.f90

```
subroutine fylgni(S,l,sum1)
use parameter
implicit none

real (kind=8)::sum1

integer,dimension(Nx,Ny)::S

integer::l,m,n,m1,m2,n1,n2


sum1=0

do m=1,Nx

   do n=1,Ny

      m1=m-1
      m2=m+1
      n1=n-1
      n2=n+1


      if (m1<=0) then

         m1=m1+Nx
         else if (m2>Nx) then
            m2=m2-Nx
         end if

         if (n1<=0) then
            n1=n1+Ny
         else if (n2>Ny) then
            n2=n2-Ny
         end if

         sum1=sum1+S(m,n)*(S(m1,n) + S(m2,n) + S(m,n1) + S(m,n2))

      end do
   end do

   sum1=sum1/(4*float(Nx*Ny))

   end subroutine fylgni
```

# Heimildir

[1] Sigurður Ingi Erlingsson, *Ising Model,* fyrirlestrarglósur úr Tölvueðlisfræði haust 2006. Slóðin er
http://rashba.raunvis.hi.is/~sie/Kennsla/isingIntro.pdf

[2] Giordano, N. J., Nakanishi, H., *Computational Physics*, Pearson Education, Inc., United States of America, 2006.