Lecture series

INTRODUCTION TO COMPUTER SIMULATION METHODS

08.01. 10:00 Ecole d'Application des
15.01. 10:00 Hautes Polymeres
23.01. 08:00 "
29.01. 10:00 "
05.02. 10:00 "

Dr. Martin Kroger
Institute of Polymers, Polymer Physics
ETH Zentrum, 8092 Zurich, Switzerland
Email: mkroeger@ifp.mat.ethz.ch
WWW: http://finch.ethz.ch/~mk

CONTENTS

LITERATURE

- Earlier lecture manuscript available
- D.W. Heermann,
  Computer simulation methods in Theoretical Physics
  Springer, Berlin, 1990
- R.B. Bird, C.F. Curtiss, R.C. Armstrong, O. Hassager
  Dynamics of polymeric liquids Vol. 1+2
  John Wiley & Suns, NY, 1987

This postscript document is available by World Wide Web.
It can be downloaded and therefore printed at any time.
Address: http://finch.ethz.ch/~mk/vorlesung_SB98.sc.ps

FORTRAN AND DATA HANDLING

- compilers (creating executable a.out)

```
UNIX machines: f77 [or f90] file.f
Microsoft Fortran: fl file.for
```

- editors

```
UNIX: vi, jot, emacs
```

- manual pages for a >command<

```
UNIX: man >command<   (eg: man f77)
```

- change directory

```
      cd >directory_name<
up:   cd ..
home: cd
```

- list files

```
UNIX: ls, ls -l (long), ls -lt (sorted by time)
```

- move or copy file

```
UNIX: mv/cp file newfile
DOS:  move/copy file newfile or Norton commander
Windows: Data-Manager
```

- print file

```
UNIX : lp filename (print); lpstat (status)
       lpr filename (print); lpq (status)
       (specify printer name: see man lp/lpr)
```

- generate postscript file, analyse data

```
UNIX: xmgr, showcase, framemaker
Windows: Origin
```

- view postscript file

```
UNIX: ghostview, xpsview, xv, acroread
```

- search string in file

```
UNIX: grep string filename
```

- stop execution of program: crtl c

INTRODUCTION TO FORTRAN PROGRAMMING LANGUAGE

1) Problem: ask for an integer value, return the faculty
Solution:

```
* ---------------------------------------------------------------------
                                ! comments:

      program faculty           ! Name of program
      integer value,faculty     ! Variablen declaration:
                                ! e.g. integer,real,double precision

      print *, ' Value: ',      ! print on screen
                                ! (* stands for Format: *=arbitrary)
      read *, value             ! read from keyboard
      facultaet=1
      do i=1,value              ! Loop, i takes values 1,2,.., value
       faculty=faculty*i
      end do                    ! End of loop
      print *, ' Result: ',faculty   ! Output to screen

      print 10,'Result', faculty     ! Formatted output
                                     ! (I6=integer with 6 numbers)
10    format (A10,I6)                ! A10=Text with 10 characters


      stop           ! stop of exution
      end            ! obligatory
* ---------------------------------------------------------------------
```

Note: Faculty(0)=1 (loop feature, may depend on compiler option)

MONTE CARLO METHODS / BROWNIAN DYNAMICS

need

Uniform Random number generators

- **Silicon Graphics**

```
double precision rand
random=rand()
```

- **SUN**

```
random=rand(0)
```

- **CRAY**

```
random=ranf()
```

- **Microsoft Fortran**

```
random=0.xxx

random=rand_(random)
...

function rand_(orandom)
parameter (modulus=2**32-1,ia=16807)
rmodulus=2.**32.-1.
ix=nint(orandom*rmodulus)
ix=ix*ia
if (ix.lt.0) ix=ix+modulus+1
rand_=real(ix)/rmodulus
return
end
```

EXERCISES

```
* ------------------------------------------------------------------------

2) Problem: Edit the program, compile and run
   Solution: UNIX: login, vi filename.f, f77 filename.f, a.out
             Microsoft: edit filename.for, fl filename, a.out

* ------------------------------------------------------------------------

3) Problem: as 1) but allow for multiple questions
            and use a subroutine tp program it
    Solution:
* ------------------------------------------------------------------------
       program facultaet
       integer value

1      continue                   ! set marker '1'
       print *, ' Value: ',
       read *, value
       if (value.lt.0) goto 2     ! compare values (lt=less than)

       call facul(value)          ! call a subroutine with parameter
       goto 1                     ! goto marker

2      continue
       print *, ' Stop code since value < 0.'
       stop
       end

       subroutine facul(value)    ! Name of subroutine
       integer value,facultaet

       facultaet=1
       do i=1,value
        facultaet=facultaet*i
       end do
       print *, ' Result: ',facultaet

       return                               ! leave subroutine
       end                                  ! obligatory
* ------------------------------------------------------------------------
```

EXERCISES

4) Problem: Create a sequence of random numbers (in 0,1)
Solution:

```
* -------------------------------------------------------------------------
      program random_number
      integer sequenz
      real random,orandom

      print *, 'initial value for random generator? (eg. 0.3)'
      read *, random
      print *, 'Length of sequence?'
      read *, sequenz

      do i=1,sequenz
       random=rand_(random)
       print *, i, random
      end do

      function rand_(orandom)
      real rmodulus,orandom
      integer ix,ia,modulus
      parameter (modulus=2**32-1,ia=16807) ! fixed parameter
      rmodulus=2.**32-1.          ! 2.**32  = 2.0^32  ! Dots important!
      ix=nint(orandom*rmodulus)   ! nint(x) = next integer value
      ix=ix*ia
      if (ix.lt.0) ix=ix+modulus+1
      rand_=real(ix)/rmodulus     ! real(x) = convert integer -> real
      return
      end
```

EXERCISES

5) Problem: as 4), add average and variance of random number sequence
   Solution:

```
* ---------------------------------------------------------------------
      program random_number
      integer sequenz
      real random,orandom,average,standard

      print *, 'initial value for random generator? (eg. 0.3)'
      read *, random
      print *, 'Length of sequence?'
      read *, sequenz

      average=0.          ! initialize values
      standard=0.         !

      do i=1,sequenz
       random=rand_(random)
       average=average+random          ! accumulate
       standard=standard+random**2     !
      end do

      average=average/real(sequenz)    ! Normalization
      standard=standard-average**2     !
      standard=sqrt(standard/real(sequenz))   ! Definition

      print *, ' Length of sequence: ', sequenz
      print *, ' Average     : ', average
      print *, ' Standard Dev: ', standard

      stop
      end

      function rand_(orandom)
      ... (as in Problem 4)
* ---------------------------------------------------------------------
```

Check: Run for different sequence lengths,
       average must converge to 0.5

Other random number generators
- built in (normally)
- see 'Numerical Recipes in Fortran', Cambridge Univ. Press

EXERCISES

8) Problem: Drop N=64 disks with radius R into an area (LxL), avoid overlapping, measure the smallest distance, visualize it

Solution:

```
* ----------------------------------------------------------------------------
      problem box
      parameter(N=64)              ! Number of disks = N
      real position, laenge, radius, newpos
      real xcoord, ycoord, xij, yij, rij, min_distance
      real density, abstand
      dimension position(N,2)      ! disk coordinates (2D field)
      print *, ' Box size: '
      read *, box
      print *, ' Radius of disks: ',
      read *, radius
      print *, ' Init random number: '
      read *, random

      min_distance=1000.           ! Inititalisation > max. distance

      do i=1,N                     ! Loop for all disks
* Add disk i to random position, if this position is free
1      continue
       random=rand_(random)
       xcoord=random               ! random x-coordinate in (0,box)
       xcoord=xcoord*box
       random=rand_(random)
       ycoord=random               ! random y-Koordinate in (0,box)
       ycoord=ycoord*box
* calculate distance to existing disks 1,2,..,i-1
       abstand=1000.
       do j=1,i-1
        xij=xcoord-position(j,1)   ! difference x-coordinates
        yij=ycoord-position(j,2)   !            y-coordinates
        rij=sqrt(xij**2+yij**2)! euklidian distance
        abstand=amin1(abstand,rij) ! amin1(x,y)=minimum of x,y
       end do
* reject coordinates
       if (abstand.lt.radius) goto 1 ! lt=less than (try again ..)

* accept coordinate, message to screen
       print '(A,I6,A)', ' Disk #',i,' has been added.'
* min distance between (accepted) disks
       min_distance=amin1(min_distance,abstand)
      end do

       print *, ' Radius       : ',radius
       print *, ' Area         : ',laenge**2
       density= 3.141*radius**2*real(N)/laenge**2
       print *, ' density      : ', density
       print *, ' min. distance: ',min_distance

      stop
      end

      function rand_(orandom)  ... (as above)
```

EXERCISES

MONTE CARLO SIMULATION OF THE X-Y MODEL

```
C X-Y model with periodical condition.
C The value of magnetic field H
C should be much smaller compared to the exchange coupling J.
C The linear size of the lattice is set to be 40.
C Modify PARAMETER to change it.
        PARAMETER(L=40,PI=3.14159)
        INTEGER IP(L),IM(L)
        REAL PHI(L,L),JKT,M
        WRITE(*,*)'Magnetic field H=? (eg. 0.1)'
        READ(*,*) H
        WRITE(*,*)'Parameter JKT (exchange interaction/kT)=? (eg. 0.5)'
        READ(*,*) JKT
        WRITE(*,*)'How many transient MCsteps/site to be discarded?'
        READ(*,*) N0
        WRITE(*,*)'How many MCsteps/site for data?'
        READ(*,*) MCSTEP
        PRINT *, ' -> Total number of steps: ',MCSTEP*N0
        WRITE(*,*)'initial value for random generator? (eg. 0.3)'
        READ(*,*) RANDOM
* mathcad (2 dimensional LxL field)
        if1=22
        open (if1,file='mcout.cad',form='formatted')
        rewind (if1)
C Look-up-table for periodical boundary condition.
        DO 10 I=1,L
          IP(I)=I+1
          IM(I)=I-1
10      CONTINUE
        IP(L)=1
        IM(1)=L
C Initialize the lattice.
        DO 30 I=1,L
          DO 30 J=1,L
           PHI(I,J)=0.5
30      CONTINUE
        A=1./(L*L)

...
```

EXERCISES

```
C Monte-Carlo part.
        MCSMAX=N0+MCSTEP
        M=-L*L
        DO 100 MCS=1,MCSMAX
        DO 103 K=1,N0
          DO 50 I=1,L
            DO 50 J=1,L
              ANGLE=PHI(I,J)
              RANDOM=RAND_(RANDOM)
              AGNEW=PI*(RANDOM+0.5)
              UOLD=COS(ANGLE-PHI(IP(I),J))+COS(ANGLE-PHI(IM(I),J))
     1            +COS(ANGLE-PHI(I,IP(J)))+COS(ANGLE-PHI(I,IM(J)))
              UNEW=COS(AGNEW-PHI(IP(I),J))+COS(AGNEW-PHI(IM(I),J))
     1            +COS(AGNEW-PHI(I,IP(J)))+COS(AGNEW-PHI(I,IM(J)))
              DM=COS(AGNEW)-COS(ANGLE)
              DELTA=UNEW-UOLD+H*DM
              RANDOM=RAND_(RANDOM)
              IF(DELTA.GE.0..OR.RANDOM+0.5.LE.EXP(JKT*DELTA))THEN
                PHI(I,J)=AGNEW
                IF(DELTA.GE.0) THEN
c                 print '(I3,3F10.3)', 1, DELTA, UOLD, UNEW
                  IZA = IZA+1
                ELSE
c                 print '(I3,3F10.3)', 0, DELTA, UOLD, UNEW
                  IZB = IZB+1
                END IF
                M=M+DM
              END IF
50        CONTINUE
103     CONTINUE
            IF(MCS.EQ.N0) WRITE(*,101) N0,A*M
            IF(MCS.GT.N0) WRITE(*,102) MCS-N0,A*M
100     CONTINUE
* cad format
      WRITE(if1,'((6(E9.3,1X)))') ((phi(i,j),j=1,L),i=1,L)
      print *, ' mcout.cad file created.'
      close (if1)
101     FORMAT(1X,'After N0=',I3,' transient MCsteps/site, M=',F5.2)
102     FORMAT(1X,'After',I4,' MCsteps/site, M=',F5.2)
        END
```

(take function RAND_ from above)

EXERCISES

MONTE CARLO METHOD TO PRODUCE A SEQUENCE OF STATES
ACCORDING TO A GIVEN PROBABILITY FUNCTION

```
      program mc_gen_arbitrary_1Dfunction
* generation of 1D states according to given probabilty function
* via Monte Carlo method (Metropolis algorithm)
* positive function fkt has to be defined (off-lattice)
* possible states on a lattice with dimension: lat (parameter)

* settings:  define fkt below
* lattice parameter: lat
      parameter(lat=100)
      double precision rand
      real rho(lat)
      common /dat/ rho

      nsteps=10000000
      rlat=real(lat)
      call initrho

      print *, ' -----------------'
      do i=1,lat
       print *, i, fkt(i)
      end do
      print *, ' -----------------'

* select init position
      irandom=int(rand()*rlat)+1
* loop
      do loop=1,nsteps
* select new 'particle' jrandom
       jrandom=int(rand()*rlat)+1
* calculate functional values
       rfkti=fkt(irandom)
       rfktj=fkt(jrandom)
       if (rfktj.ge.rfkti) then     ! accept state
        irandom=jrandom
       else
        if (rand().le.rfktj/rfkti) then  ! accept state
         irandom=jrandom
        end if
       end if
      rho(irandom)=rho(irandom)+1.    ! COUNT STATE HERE !!
      if (mod(loop,nsteps/20).eq.0) call stddev(loop)
      end do

      call normalize_rho

      stop
      end
```

... continuation

```
      function fkt(i)
      real U
       ri=real(i)/50.     ! Note: i=1,..,lat
       U=ri**2-ri**3
       fkt=exp(-U)
       if (fkt.lt.0.) stop ' cannot be handled'
      return
      end

      subroutine normalize_rho
      parameter(lat=100)
      real rho(lat)
      common /dat/ rho
      common /ave/ rave
       do i=1,lat
        rho(i)=rho(i)*rave
        print *, i, fkt(i), rho(i)
       end do
      return
      end

      subroutine initrho
      parameter(lat=100)
      real rho(lat)
      common /dat/ rho
      do i=1,lat
       rho(i)=0.
      end do
      return
      end

      subroutine stddev(loop)
      parameter(lat=100)
      real rho(lat)
      common /dat/ rho
      common /ave/ rave
      rave=0.
      rstd=0.
      do i=1,lat
       rhelp=fkt(i)/rho(i)
       rave=rave+rhelp
       rstd=rstd+rhelp**2
      end do
      rave=rave/real(lat)
      rstd=rstd/real(lat)
      rstd=sqrt(rstd-rave**2)
      print *, loop, rstd
      return
      end
```