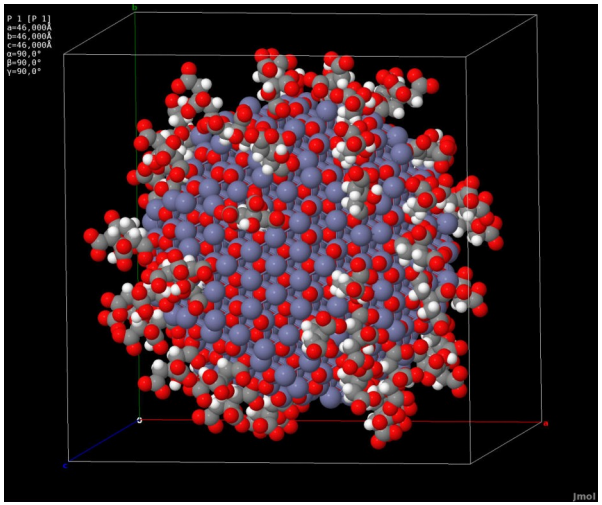
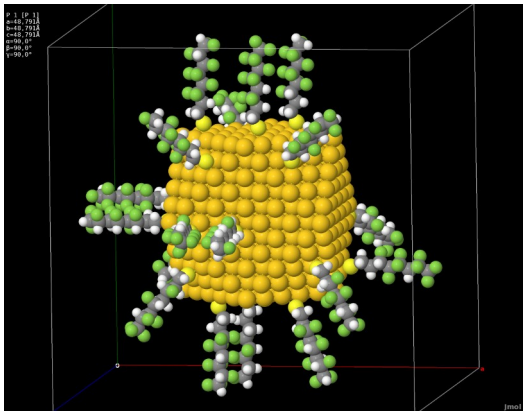
tutorial session IX

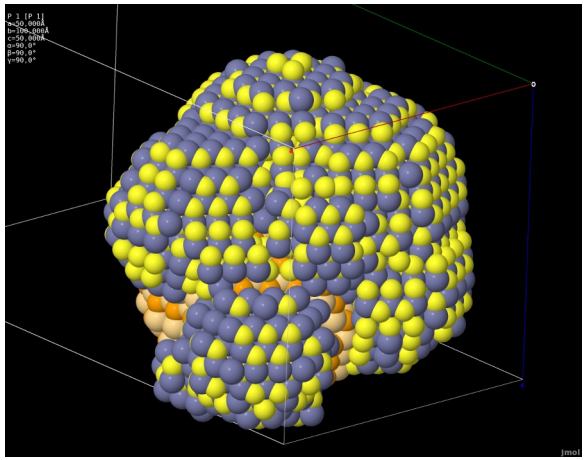
Nanoparticle refinement



ZnO
with organic ligand



Gold cuboctahedron
with organic ligands



CdSe / ZnS
core / shell

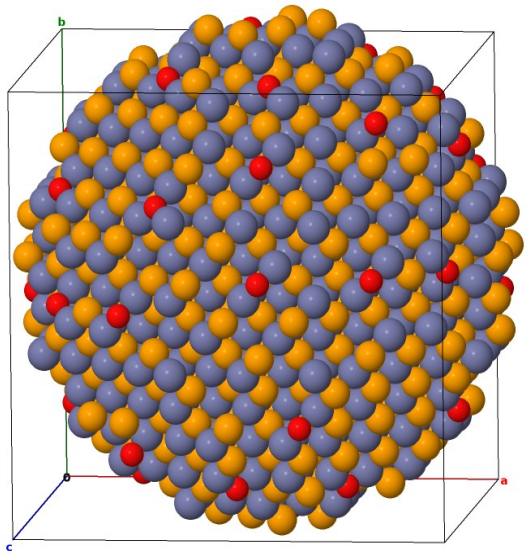
Nanoparticle refinement

Goal: ellipsoidal ZnSe nanoparticle with
Stacking faults
Refine against experimental PDF

Concept: DIFFEV:
Define population
Define parameters and allowed range
Loop over generations

DISCUS
Build an ellipsoidal NP
Calculate PDF

KUPLOT
Average
Shape corrections
Calculate R-value



Start discus_suite
Select directory Lectures\09.Nanoparticle_Refinement\ZNSE.PDF_suite_mpi

suite> **@refine.mac**

start second discus_suite

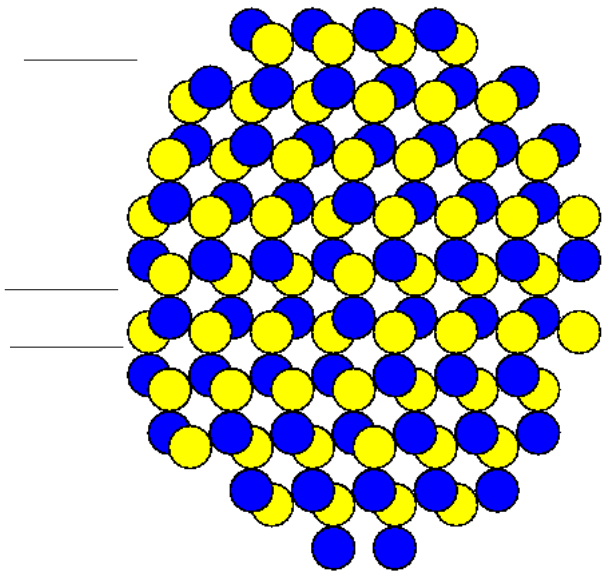
```
set prompt,off                    ! Allow for prettier output
#
diffev              ! switch to DIFFEV section
   @diffev.mac      ! run the real refinement macro
exit                ! finish discus_suite
```

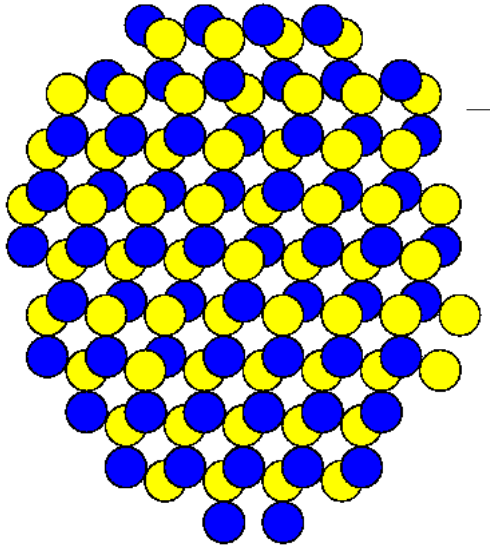
Essentially just a reminessence to former times,
Could all be integrated into macro diffev.mac … to be done in future releases…

The essential refinement macro

```
diffev                  ! switch to DIFFEV section
@cleanup.mac            ! discard previous results !!!!!
#
REF_NINDIV = 5          ! define number of individual repetitions
@get_model.mac          ! define the refinement model
@diffev_setup.mac       ! define all diffev essentials
#
init silent             ! create the first generation
#
do i[0]=1,500           ! run for a fixed number of cycles/generations
   echo „Generation %5d“,REF_GENERATION  ! keep user informed
   run_mpi discus, dis.diffev.mac, repeat:REF_NINDIV, compute:serial,
                   logfile:LOGFILES/d ! Do magic
   compare silent       ! Compare parent and child generation
enddo                   ! End of loop indicator
exit                    ! Return to suite
```

Aspects of PDF calculation for small nanoparticles

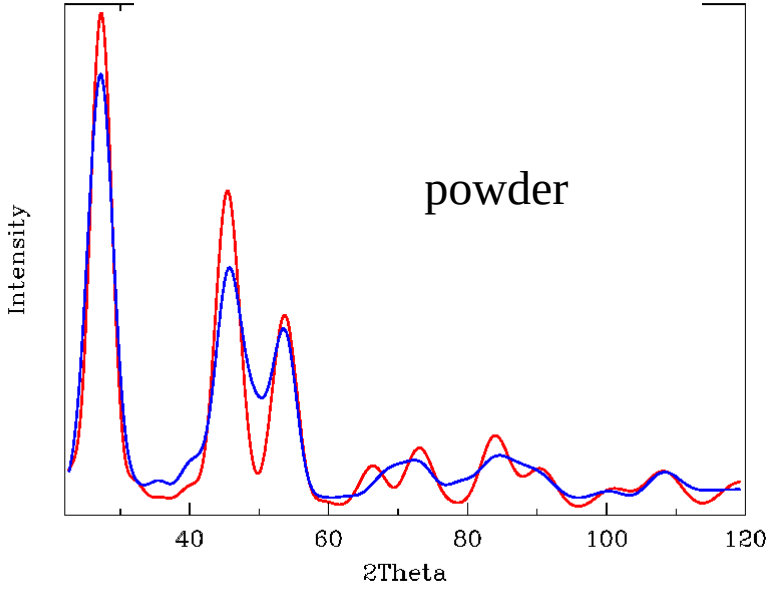


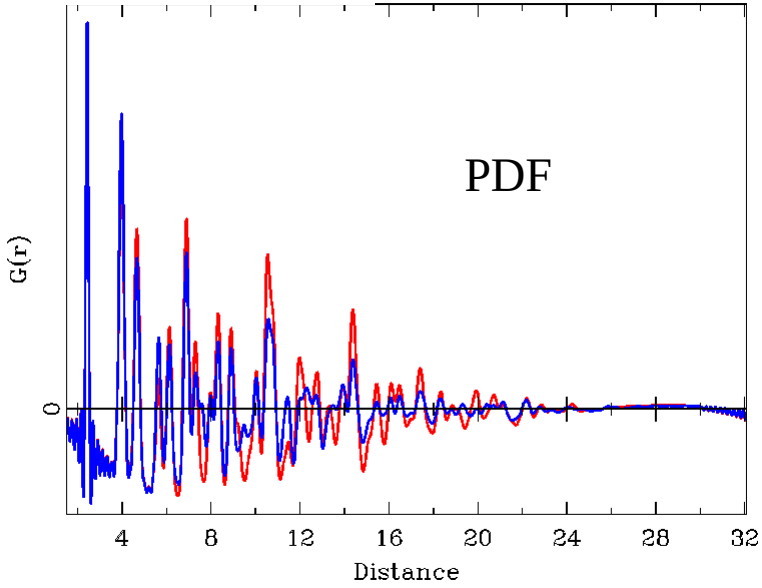


10 layers
simulated with
identical parameters

individual location
of stacking faults

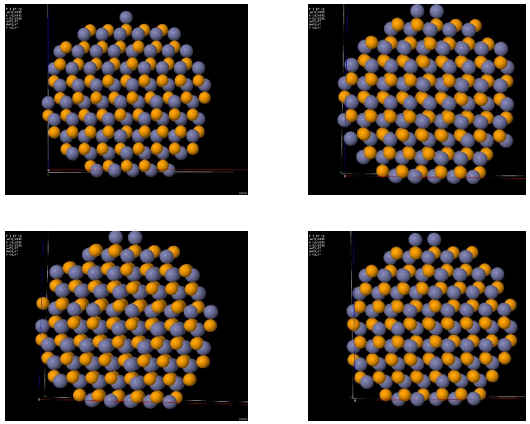
each particle is not
representative
==> need to average

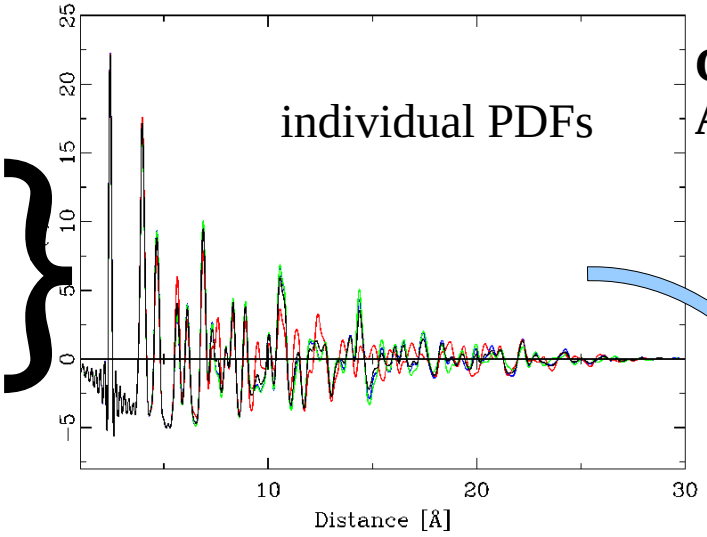




Bottom-Up Simulation and Refinement

Ensemble modelling

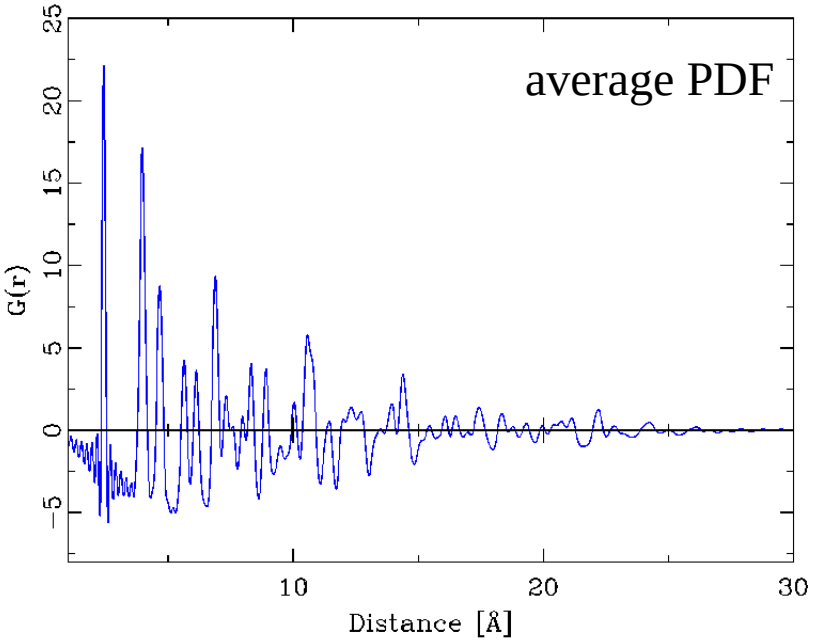




Calculate (many) individual nanos
Average PDF / powder pattern

coordinates in asymmetric unit
symmetry
diameter
defects

2 lattice parameters
1 coordinate
1 ADP
2 radii
1 probability

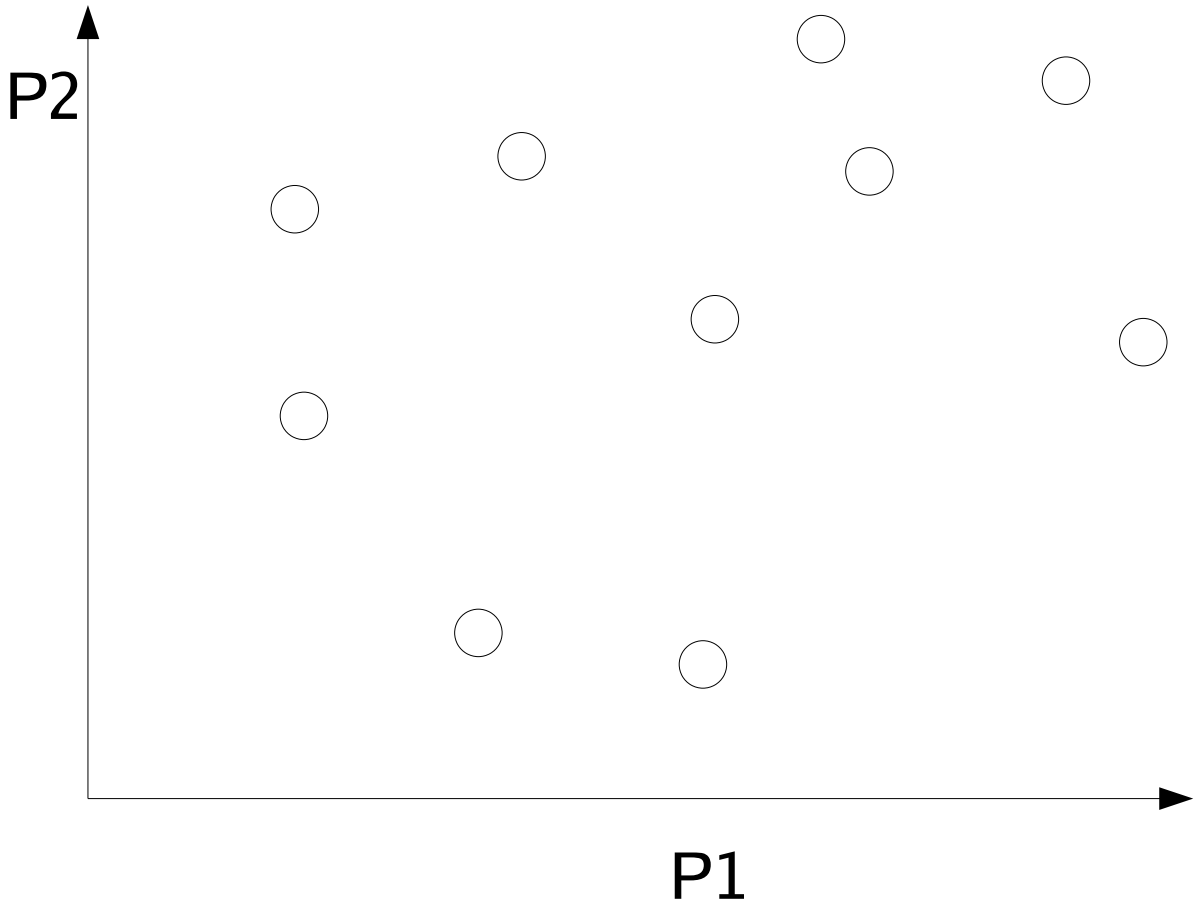


BUT

incoherent average of PDFs
requires evolutionary refinement
no Least-Squares

expensive

Niederdraenk et al. Phys.stat.sol c (2007);
Niederdraenk et al. PCCP (2011)



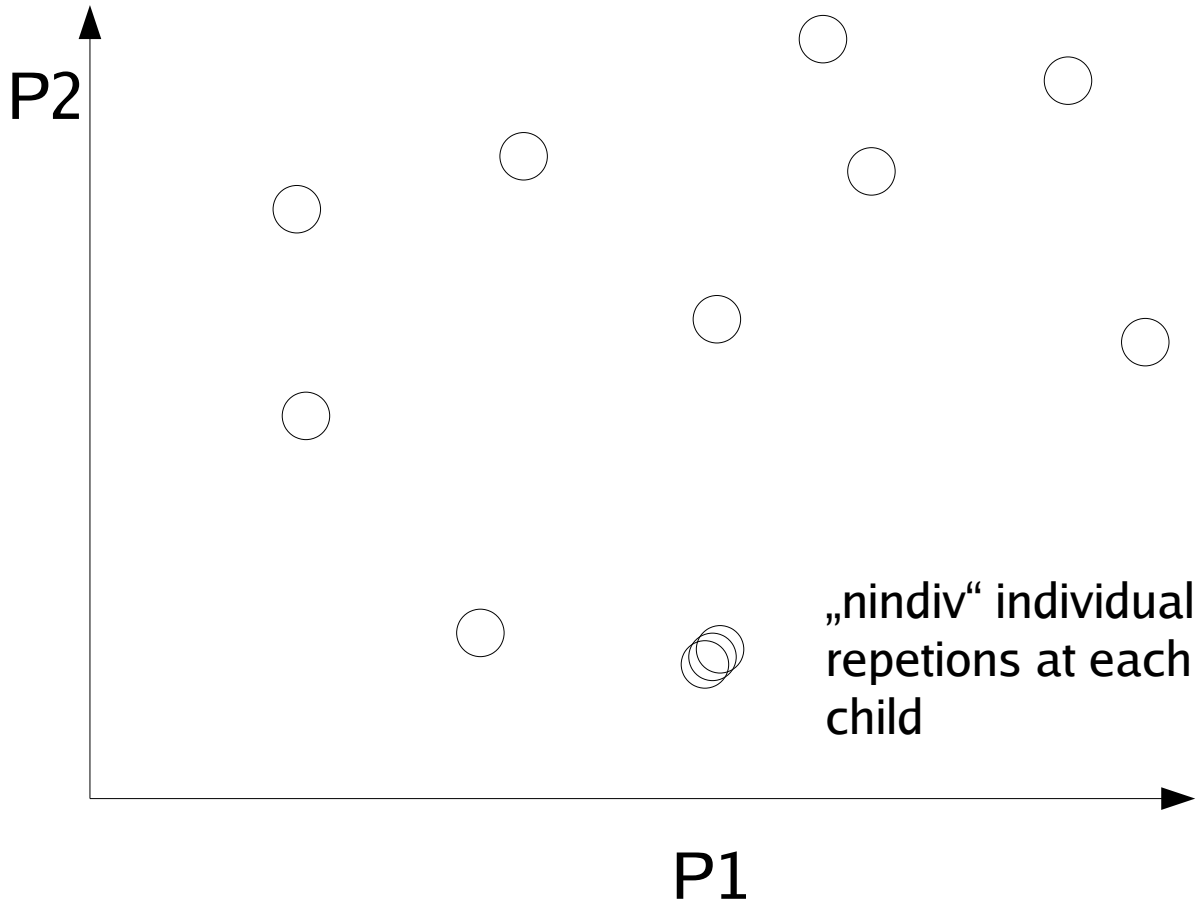
$y = P_1x + P_2$

N sets of parameters
(P_1, P_2)

R-values for each set

Each parameter vector
Represents a nanoparticle
With:
Lattice parameters a,b,c,…
Atom positions…
Shape…
Defects...

Diffev calls these
„children, kids“



$y = P_1x + P_2$

N sets of parameters
(P_1, P_2)

R-values for each set

Each parameter vector
Represents a nanoparticle
With:
Lattice parameters a,b,c,…
Atom positions…
Shape…
Defects...

Diffev calls these
„children, kids“

The essential refinement macro

```
diffev                  ! switch to DIFFEV section
@cleanup.mac            ! discard previous results !!!!!
#
REF_NINDIV = 5          ! define number of individual repetitions
@get_model.mac          ! define the refinement model
@diffev_setup.mac       ! define all diffev essentials
#
init silent             ! create the first generation
#
do i[0]=1,500           ! run for a fixed number of cycles/generations
   echo „Generation %5d“,REF_GENERATION  ! keep user informed
   run_mpi discus, dis.diffev.mac, repeat:REF_NINDIV, compute:serial,
                   logfile:LOGFILES/d ! Do magic
   compare silent       ! Compare parent and child generation
enddo                   ! End of loop indicator
exit                    ! Return to suite
```

```
variable  integer,ipar  ! define a dummy variable
#
pop_gen[1]  =   0       ! Make this the generation ZERO
pop_n[1]    =  10       ! we have 10 members in the population
pop_c[1]    =  10       ! we have 10 children in the population
pop_dimx[1] =  14       ! There will be 14 refined parameters
#
#  1 lattice constant a
#
newpara P_lata,  3.90, 4.02, 3.90, 4.02
#  ! With newpara I define for each parameter:
#
#  ! P_lata:  a character string that names the parameter
#  ! 3.900:   a hard lower boundary
#  ! 4.020:   a hard upper boundary
#  ! 3.980: lower boundary for the initialization in generation ZERO
#  ! 3.990: upper boundary for the initialization in generation ZERO
#
```

```
diff_cr[1]   = 0.9    ! Cross_over probability
diff_f[1]    = 0.81   ! scale factor for difference vector < 1
diff_lo[1]   = 0.0    ! Probability for „local“ refinement leave at 0
diff_k[1]    = 1.0    ! Location of donor base, leave at 1
#
refine       none     ! Initially fix all parameters
refine       P_lata, P_latc, ... ! Refine these
#
donor       random    ! randomly choose donor, instead of „best“ member
#selection   compare  ! Stricly compare parent and its child
selection   best,all  ! Take better half of all parents and children
#
trialfile   silent    ! Do not write trial files to disk to save I/O
restrial   silent     ! Do not read R-values from disk, transfer internally
logfile     DIFFEV/Parameter    ! Complete archive of the refinement
summary    DIFFEV/Summary       ! Shorter summary of refinement
lastfile    DIFFEV/Current      ! Complete info just on the last generation
#
backup TEMP/calc, FINAL/final ! Automatically backup TEMP/calc.0001 etc.
```

The essential refinement macro

```
@cleanup.mac              ! discard previous results !!!!!
REF_NINDIV = 5            ! define number of individual repetitions
@diffev_setup.mac         ! define all diffev essentials
init silent               ! create the first generation
do i[0]=1,500             ! run for a fixed number of cycles/generations
   run_mpi discus, dis.diffev.mac, repeat:REF_NINDIV, compute:serial,
                   logfile:LOGFILES/d ! Do magic
   compare silent         ! Compare parent and child generation
enddo                     ! End of loop indicator
```

run_mpi command

Instructs diffev to switch internally to „discus“ section and to execute macro „dis.diffev.mac“ I use this name as a generic interface between diffev and discus

LOGFILES/d Copy output to files d.0001 etc in LOGFILES (parallel only)

The interface macro; generic and suitable for *allmost all* refinements

```
set error,exit                    ! Stop calculation if an error occur
variable integer,indiv            ! Variable for local repetitions
branch kuplot                     ! Switch to KUPLOT section
   reset                          ! Within the suite, diffev reserves these
exit                              ! Entries in global i[] variable
#
do indiv = 1, REF_NINDIV          ! Loop over all individual repetitions
  @discus.znse.mac kid            ! The specific macro for our refinement
enddo                             ! End of loop indicator
#
branch   kuplot                   ! Switch to KUPLOT
   @kup.diffev.mac ., kid         ! Execute the main kuplot macro
#
exit                              ! Return to diffev
```

The macro with specific instructions, changes for each refinement, shortened

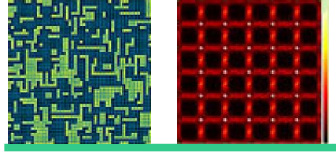
```
@variables.znse.mac ! variable names are easier to remember
read                ! create a template incl Space group symbol
   free P_lata, P_lata, P_latc, 90.00, 90.00, 120.00, P63mc
insert Zn, 1./3., 2./3., P_z_zn, P_biso ! Insert atoms at ideal positions
insert Se, 1./3., 2./3., 0.0000, P_biso ! Does not need unnecessary I/O
#
save                ! Save the changed asym.unit IMPORTANT: unique name
  outfile "%c/STRU/znse_wurtzite.%4D.%4D.cell", TMPDIR,REF_KID,indiv
  run
exit
@makelayers.mac znse_wurtzite      ! Prepare layer to stack
@shape.ellipsoid.mac znse_wurtzite ! Create the actual nanoparticle
@pdf.mac REF_KID,indiv  ! Calculate and save PDF (or powder pattern...)
```

```
set error,exit                    ! Stop calculation if an error occurs
@global.mac                       ! Directory names
@get_model.mac                    ! Ensure we have correct value of NINDIV
#                                 ! For parallel version on HPC
fexist "%c/DATA/%c.%4D", DATADIR, DATAFILE, REF_KID
if(res[1].eq.0) then  ! File does not exist load up to DATADIR
   system "cp DATA/%c %c/DATA/%c.%4D", DATAFILE, DATADIR, DATAFILE, REF_KID
endif
@kup.average.mac                  ! Merge all individual calculations   nindiv + 1
load xy, "%c/DATA/%c.%4D", DATADIR, DATAFILE, REF_KID  !                nindiv + 2
spline REF_NINDIV+1, REF_NINDIV +2 ! Ensure identical x-axis-scale nindiv + 3
@kup.fit.polynomial.mac           ! Correct 4PI RHO line and scale      nindiv + 7
rval REF_NINDIV+2, REF_NINDIV+7, dat ! calc R-value, transfered internally
reset                             ! No DATA
exit                              ! Back to DISCUS / DIFFEV (depends on use)
```

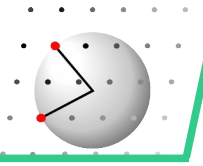
```
#
if(n[1]==0) then
# alternative version if individual calculations were written to disk
  reset
  do indiv=1,REF_NINDIV
     load xy,"%c/INDI/indi.%4D.%4D", INDIDIR, REF_KID,indiv
  enddo
endif
#
merge all     ! creates new data set number: nindiv + 1

# Often several phases are merged, this requires modification of
# REF_NINDIV
# REF_NINDIV = REF_NINDIV + 1
```

If more than one phase was calculated in DISCUS, REF_NINDIV might need to be adjusted
Or if the temporary individual PDF's were written to disk they need to be read here.

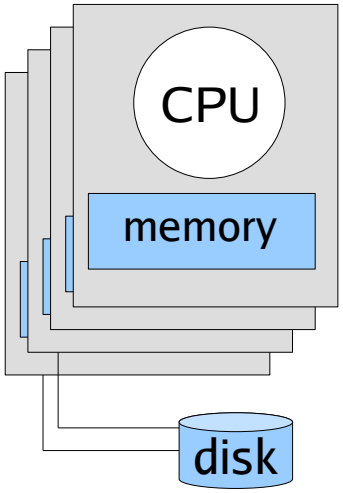
```
kcal sub,REF_NINDIV+2,REF_NINDIV+3 ! Creates data set nindiv + 4
skal                               ! Fit needs to know x and y range
fit REF_NINDIV + 4                 ! Fit a function to data set nindiv + 4
   func, poly,6                    ! Polynomial of order x^6
   para 1,0, 0.00                  ! x^0 is fixed to 0.00
   para 2,1, 1.00                  ! x^1 refined, starts at 1.0
   para 3,1, 0.00                  ! number, flag, starting_value
   para 4,1, 0.00
   para 5,1, 0.00
   para 6,1, 0.00
   para 7,1, 0.00                  ! x^6 refined, starts at 0.00
   wic  dat                        ! Use weights from data set
   cycle 200                       ! 200 cycles should be enough
   urf 0.5                         ! control refinement speed ~1/urf
   run                             ! Let's do it
 exit                              ! back to main KUPLOT menu
 kcal add,REF_NINDIV+3,REF_NINDIV + 5 ! Add polynomial to merged PDF's
 ksav REF_NINDIV + 7               ! Save this calculated PDF
    outf "TEMP/calc.%4D",REF_KID
    run                            ! automatically returns to main menue
```



tutorial session X
parallel
(Nanoparticle)
refinement



Multicore node

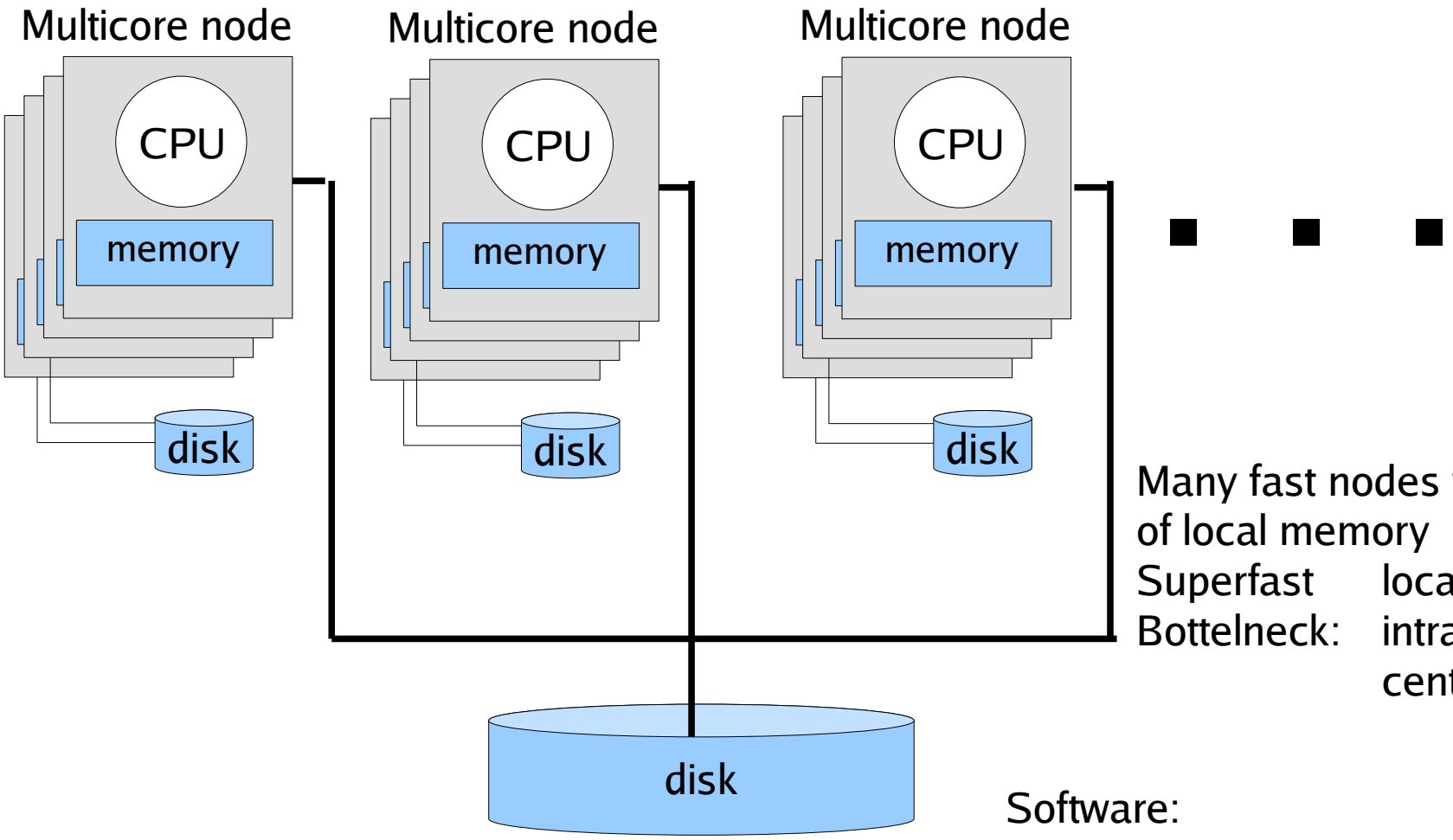


One nodes with
no local memory
fast local disk

Single user PC
1 nodes, each with 4 cores

Software:
MPI Message Passing Interface
OpenMP
and combinations
hybrid programming

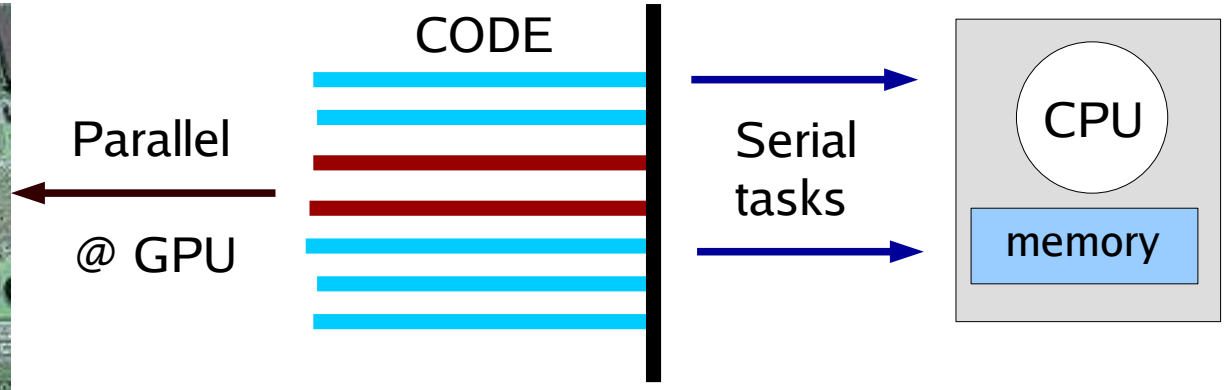
Parallel refinement, Architecture



Many fast nodes with lots of local memory
Superfast local disk
Bottelneck: intranet to central disk

Emmy Cluster RRZE, Erlangen
500 nodes, each with 10 cores

Software:
MPI Message Passing Interface
OpenMP
and combinations
hybrid programming



GPU

Thousands of cores
little local memory

Software:

Proprietary CUDA
OpenCL
GPUOpen
???

Message passing Interface

```
Easy installation on Linux, Cygwin ! Install like any other package
#
mpiexec -n 5 discus_suite -macro refine.mac ! discus_suite started indirectly
      ! -n 5 : run 5 tasks in parallel
```

Windows: start a regular Discus_suite window

```
# Easy start: ! Just type
#
suite > parallel refine.mac ! discus_suite started indirectly
              ! automatically get all cores
```

Required changes in ZNSE macros: None !

The essential refinement macro

```
@cleanup.mac          ! discard previous results !!!!!
#
@setup.mac            ! define number of individual repetitions
@diffev_setup.mac     ! define all diffev essentials
#
init silent           ! create the first generation
#
do i[0]=1,500         ! run for a fixed number of cycles/generations
   echo „Generation %5d“,pop_gen[0]  ! keep user informed
   run_mpi discus, nano.znse.mac, repeat:REV_NINDIV, compute:serial,
                   logfile:lOGFILES/d ! Do magic
   compare silent     ! Compare parent and child generation
enddo                 ! End of loop indicator
exit                  ! Return to suite
```