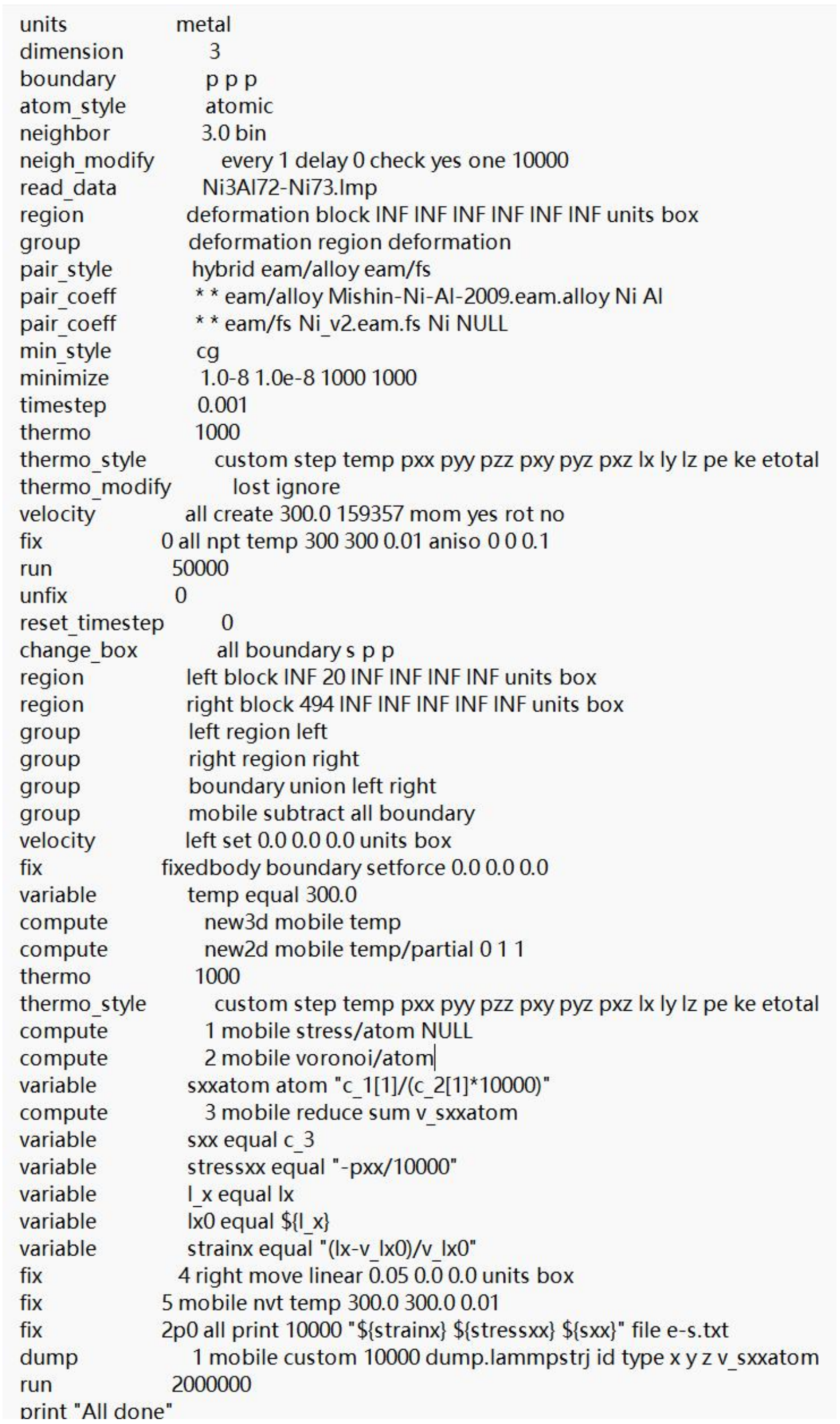
Data availability statements

Due to the large amount of data related to the calculation results in this paper and the limitation of the data sharing conditions of our institution, it is not possible to upload all the data to the public network, so we disclose the relevant calculation script files involved in the research process.

The molecular dynamics simulation software for Large-scale Atomic/Molecular Massively Parallel Simulator (LAMMPS) can be found at https://www.lammps.org/. The version of the software employed for this study is version [LAMMPS (29 Sep 2021 - Update 3)].

The molecular dynamics simulation script of the LAMMPS software as follow:

the input script file

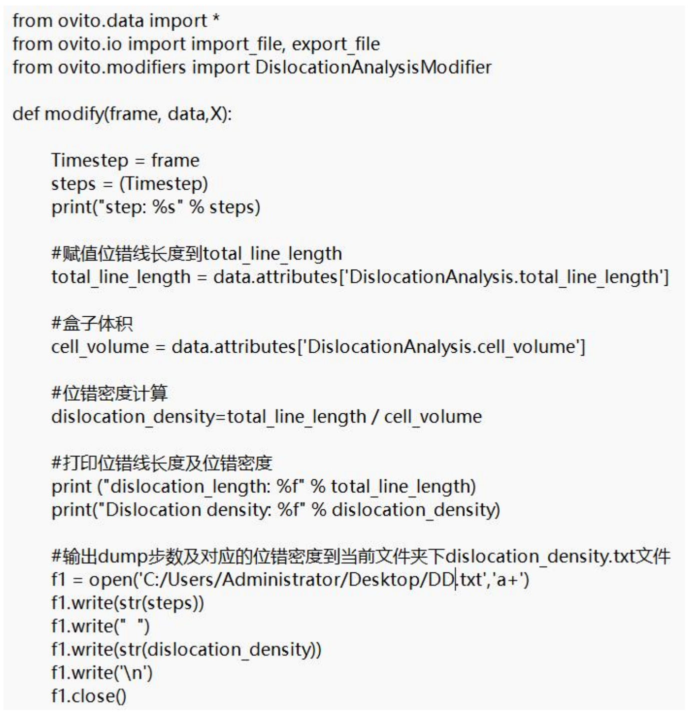


```
units              metal
dimension            3
boundary             p p p
atom_style           atomic
neighbor             3.0 bin
neigh_modify           every 1 delay 0 check yes one 10000
read_data            Ni3Al72-Ni73.lmp
region             deformation block INF INF INF INF INF INF units box
group              deformation region deformation
pair_style          hybrid eam/alloy eam/fs
pair_coeff          * * eam/alloy Mishin-Ni-Al-2009.eam.alloy Ni Al
pair_coeff          * * eam/fs Ni_v2.eam.fs Ni NULL
min_style           cg
minimize             1.0-8 1.0e-8 1000 1000
timestep            0.001
thermo              1000
thermo_style           custom step temp pxx pyy pzz pxy pyz pxz lx ly lz pe ke etotal
thermo_modify           lost ignore
velocity           all create 300.0 159357 mom yes rot no
fix             0 all npt temp 300 300 0.01 aniso 0 0 0.1
run               50000
unfix             0
reset_timestep          0
change_box            all boundary s p p
region             left block INF 20 INF INF INF INF units box
region             right block 494 INF INF INF INF INF units box
group              left region left
group              right region right
group              boundary union left right
group              mobile subtract all boundary
velocity           left set 0.0 0.0 0.0 units box
fix             fixedbody boundary setforce 0.0 0.0 0.0
variable            temp equal 300.0
compute              new3d mobile temp
compute              new2d mobile temp/partial 0 1 1
thermo              1000
thermo_style           custom step temp pxx pyy pzz pxy pyz pxz lx ly lz pe ke etotal
compute              1 mobile stress/atom NULL
compute              2 mobile voronoi/atom
variable            sxxatom atom "c_1[1]/(c_2[1]*10000)"
compute              3 mobile reduce sum v_sxxatom
variable            sxx equal c_3
variable            stressxx equal "-pxx/10000"
variable            l_x equal lx
variable            lx0 equal ${l_x}
variable            strainx equal "(lx-v_lx0)/v_lx0"
fix              4 right move linear 0.05 0.0 0.0 units box
fix             5 mobile nvt temp 300.0 300.0 0.01
fix             2p0 all print 10000 "${strainx} ${stressxx} ${sxx}" file e-s.txt
dump                1 mobile custom 10000 dump.lammpstrj id type x y z v_sxxatom
run               2000000
print "All done"
```

The data analysis software for The Open Visualization Tool (OVITO) can be found at https://www.ovito.org/manual/index.html#. The version of the software employed for this study is version [OVITO 3.3.0 released].

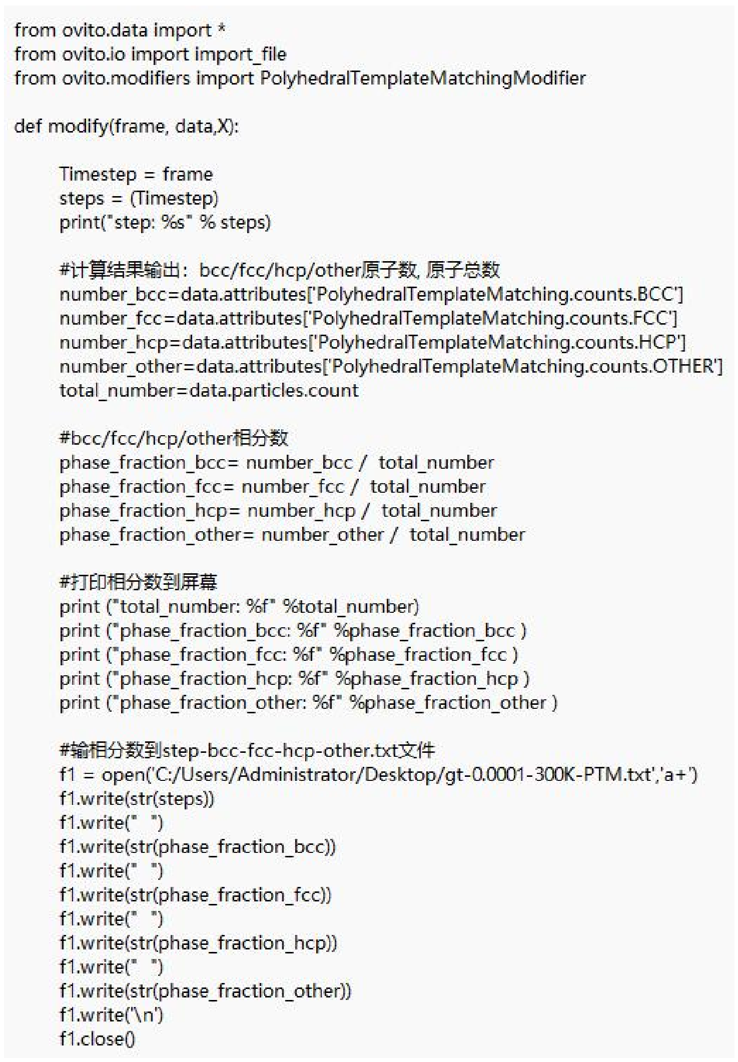
The data analysis scripts of the OVITO software as follow:

the dislocation density analysis script file



```
from ovito.data import *
from ovito.io import import_file, export_file
from ovito.modifiers import DislocationAnalysisModifier

def modify(frame, data,X):

    Timestep = frame
    steps = (Timestep)
    print("step: %s" % steps)

    #赋值位错线长度到total_line_length
    total_line_length = data.attributes['DislocationAnalysis.total_line_length']

    #盒子体积
    cell_volume = data.attributes['DislocationAnalysis.cell_volume']

    #位错密度计算
    dislocation_density=total_line_length / cell_volume

    #打印位错线长度及位错密度
    print ("dislocation_length: %f" % total_line_length)
    print("Dislocation density: %f" % dislocation_density)

    #输出dump步数及对应的位错密度到当前文件夹下dislocation_density.txt文件
    f1 = open('C:/Users/Administrator/Desktop/DD.txt','a+')
    f1.write(str(steps))
    f1.write(" ")
    f1.write(str(dislocation_density))
    f1.write('\n')
    f1.close()
```

the phase volume fraction analysis script file



```
from ovito.data import *
from ovito.io import import_file
from ovito.modifiers import PolyhedralTemplateMatchingModifier

def modify(frame, data,X):

    Timestep = frame
    steps = (Timestep)
    print("step: %s" % steps)

    #计算结果输出：bcc/fcc/hcp/other原子数, 原子总数
    number_bcc=data.attributes['PolyhedralTemplateMatching.counts.BCC']
    number_fcc=data.attributes['PolyhedralTemplateMatching.counts.FCC']
    number_hcp=data.attributes['PolyhedralTemplateMatching.counts.HCP']
    number_other=data.attributes['PolyhedralTemplateMatching.counts.OTHER']
    total_number=data.particles.count

    #bcc/fcc/hcp/other相分数
    phase_fraction_bcc= number_bcc /  total_number
    phase_fraction_fcc= number_fcc /  total_number
    phase_fraction_hcp= number_hcp /  total_number
    phase_fraction_other= number_other /  total_number

    #打印相分数到屏幕
    print ("total_number: %f" %total_number)
    print ("phase_fraction_bcc: %f" %phase_fraction_bcc )
    print ("phase_fraction_fcc: %f" %phase_fraction_fcc )
    print ("phase_fraction_hcp: %f" %phase_fraction_hcp )
    print ("phase_fraction_other: %f" %phase_fraction_other )

    #输相分数到step-bcc-fcc-hcp-other.txt文件
    f1 = open('C:/Users/Administrator/Desktop/gt-0.0001-300K-PTM.txt','a+')
    f1.write(str(steps))
    f1.write(" ")
    f1.write(str(phase_fraction_bcc))
    f1.write(" ")
    f1.write(str(phase_fraction_fcc))
    f1.write(" ")
    f1.write(str(phase_fraction_hcp))
    f1.write(" ")
    f1.write(str(phase_fraction_other))
    f1.write('\n')
    f1.close()
```