Praktikum in Rechentechniken der Physik

Runge-Kutta vierter Ordnung

Thermodynamik

von

Eduard Seifert

Sommersemester 2013

Modulverantwortlicher:
Prof. Dr. Dr. Wolfgang Cassing

JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN

Inhaltsverzeichnis

1 Runge-Kutta vierter Ordnung

Vierte Ordnung Runge-Kutta wird zum Lösen gewöhnlicher einfacher Differentialgleichungen genutzt.

$$\frac{\mathrm{d}}{\mathrm{d}x}y(x) = f(x, y) \tag{1}$$

Gewöhnliche Differentialgleichungen (DGL) höherer Ordnung können dabei auf DGL erster Ordnung reduziert werden, hierzu später mehr. Der Runge-Kutta (RK) Algorithmus integriert diese DGL näherungsweise (in diesem Beispiel nach der Zeit). Im Runge-Kutta vierter Ordnung werden dabei 4 Zwischenrechnungen ausgeführt aus denen dann der errechnete Endwert durch bestimmte Addition der Terme erhalten wird. Durch die bestimmte Addition der Zwischenrechnungen fallen die Terme erster, zweiter und dritter Ordnung weg, weshalb man einen Fehler in der Größenordnung Schrittweite[5] erhält. Daher auch RK vierter Ordnung. An vier Punkten werden die Ableitungen der Variablen ausgewertet, einmal am Startpunkt, zwei auf halber Schrittweite und einer bei voller Schrittweite. Für die Variable $y(x)$, der Schrittweite h und der Ableitung von y nach x $f(x_n, y_n)$ sieht der Algorithmus wie folgt aus:

$$\begin{aligned}
k_1 &= h\, f(x_n, y_n) \\
k_2 &= h\, f\left(x_n + \frac{h}{2}, y_n + \frac{k_1}{2}\right) \\
k_3 &= h\, f\left(x_n + \frac{h}{2}, y_n + \frac{k_2}{2}\right) \\
k_4 &= h\, f\left(x_n + h, y_n + k_3\right) \\
y_{n+1} &= y_n + \frac{1}{6}\left(k_1 + 2(k_2 + k_3) + k_4\right) + O(h^5)
\end{aligned} \tag{2}$$

Als Veranschaulichung des Algorithmus dient die folgende Abbildung (Numerical Recipes in Fortran77).

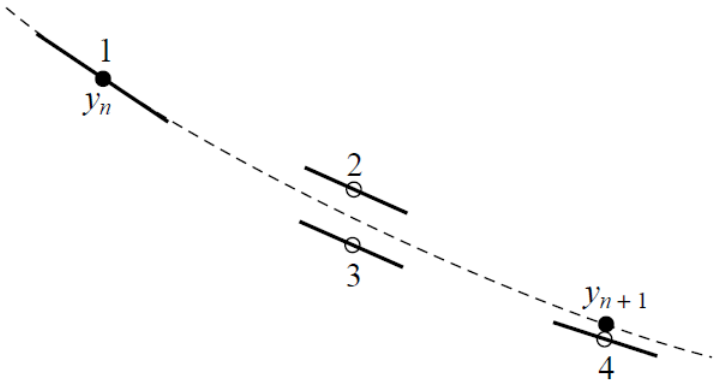


Abbildung 1: Veranschaulichung des RK vierter Ordnung: an vier Stellen werden die Ableitungen ausgewertet und nach Gleichung 2 aufaddiert.

2 Implementierung in Fortran 95

Nachdem ein Unterprogramm zur Umsetzung des RK vierter Ordnung geschrieben und im Anschluss nach dem Vorgehen von Numerical Recipes in Fortran90 optimiert wurde, wurde der nachfolgende Code zur Integration der Bewegungsgleichungen verwendet. Das Einbinden eines Moduls, in dem die Teilchenzahl steht, erspart das Übergeben der Teilchenzahl.

Listing 1: Subroutine zu Runge-Kutta vierte Ordnung.

```
!Subroutine für die Integration der DGL
!In F müssen an der i-ten Stelle die Ableitungen von y(i) nach T stehen.Die Integration
!erfolgt über die Variable T. H ist die Schrittweite in T.
SUBROUTINE RUNGEKUTTA4(N,H,T,Y,F)
IMPLICIT NONE
INTEGER, INTENT(IN)                              :: N
DOUBLE PRECISION, EXTERNAL                       :: F
DOUBLE PRECISION, DIMENSION(6*N), intent(inout) :: y
DOUBLE PRECISION, INTENT(INOUT)                  :: T
DOUBLE PRECISION, INTENT(IN)                     :: H
DOUBLE PRECISION, DIMENSION(6*N)                 :: k1,k2,k3,HF
DOUBLE PRECISION                                 :: T1,hh,h6
INTEGER                                          :: i

hh = h*0.5d0
h6 = h/6.0d0
call F(T,Y,k1)
T1 = T + hh
HF = y + hh*k1
call F(T1,HF,k2)
HF = y + hh*k2
call F(T1,HF,k3)
HF = y + k3*H
k2 = k2 + k3
T = T + H
call F(T,HF,k3)
y = y + h6 *(k1+2.0d0*k2+k3)

RETURN
END SUBROUTINE RUNGEKUTTA4
```

3 Beispielcode: Thermodynamik

Als Beispielcode zur Anwendung des Verfahrens dient ein thermodynamisches Problem. Zuerst sollte ein annehmbares Potential erstellt werden, das die effektive Wechselwirkung zwischen Atomen beschreibt. Durch den negativen Gradienten des Potentials wurde die Kraft, die auf ein Atom in einem gewissen Abstand wirkt, erhalten. Im Anschluss wurden 216 Teilchen auf ein kubisches Gitter gelegt. Die Teilchen besaßen entlang einer Kantenlänge einen Abstand von 4.3 Å zueinander, was etwa der Lage des Potentialminimums entspricht. Die für jede Koordinate eines jeden Teilchens zu lösende DGL war die klassische Newtonsche Bewegungsgleichung:

$$m\frac{\mathrm{d}^2}{\mathrm{d}t^2}x = F \tag{3}$$

Hierbei kann F von Ort und Zeit abhängen, in diesem Fall jedoch nur vom Abstand der Teilchen zueinander. Diese Gleichung lässt sich zu zwei Differentialgleichungen erster Ordnung reduzieren.

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}t}x &= \frac{p}{m}\\ \frac{\mathrm{d}}{\mathrm{d}t}p &= F\end{aligned} \tag{4}$$

Unter Einbeziehung eines impulsabhängigen Reibungsterms wurde das Programm zwischen 20000 und 30000 Iterationsschritte bei einer Schrittweite von 10^{-2} ps ausgeführt, bis der Grundzustand des Systems erreicht wurde. Den Teilchen wurden anschließend durch eine Monte Carlo Methode nach der Maxwell-Boltzmann-Verteilung die Impulse vergeben. Unter Berücksichtigung mehrerer Parameter sollte festgestellt werden, wann das System den Wechsel des Aggregatzustands von fest zu flüssig und von flüssig zu gasförmig vollführt. Der vollständige Code befindet sich in Abschnitt 3.1.

3.1 Programmcode

```
MODULE konstanten
!In diesem Modul befinden sich alle verwendeten Konstanten. a,b,s1 und s2 sind die Konstanten
!des Potentials/der Kraft. gamma ist ein Maß für die Stärke der Reibung. kb ist die Boltzmann-
!konstante in eV, pi die Kreiszahl und N die Anzahl der Teilchen
IMPLICIT NONE
SAVE
DOUBLE PRECISION, PARAMETER :: a=5.d0,b=0.27d0,s1=1.3d0,s2=3.0d0,masse=28.d-4,gamma=10.d0&
                               &,kb=8.6173324d-5,pi=3.141592653590d0
INTEGER, PARAMETER :: N = 6*6*6
END MODULE
!*****************************************************************************************
PROGRAM newtonrk4
USE konstanten
!Programm zum Lösen der Newtonschen Bewegungsgleichung für ein thermodynamisches Problem.
!In dem Vektor y(1,.....,6*N) stehen jeweils für beispielsweise dem ersten Teilchen:
!y(1)=x y(2)=y y(3)=z y(4)=p_x y(5)=p_y y(6)=p_z
!sodass im Abstand von 6 Einträgen von jedem Teilchen die Koordinaten und Impulse stehen.
!In diesem Programm werden die Impulse, nachdem der Grundzustand erreicht wurde per Monte
!Carlo über die Maxwell-Boltzmann Verteilung verteilt. Die Längeneinheit beträgt Angström (A),
!die Energieeinheit eV, Kraft eV/A, Zeit ps. die Einheit der Masse wird in eV ps^2/A^2 angege-
!ben, wobei 1u =1.036426866d-4 eV ps^2/A^2. Die Teilchen sollen eine Masse von 28u besitzen.
!Benutzt wird eine Masse von masse = 28.d-4 eV ps^2/A^2
IMPLICIT NONE
INTEGER :: i,j,k,anzahl,iostat
DOUBLE PRECISION :: zeit,temperatur,dt,zeitende,ort,energie,e0,r1,kinetische, potentielle&
                    &,abweichung,ortsbetrag,nenner2,rw,phiw,thetaw
!dt ist die Schrittweite,ort gibt die Ausdehnung des Systems, in energie steckt die kin. Ener-
!gie nach Ausführung von montecarlo. e0 gibt nach E_kin=3/2 kb*T die bei einem idealen Gas zu
!erwartete kinetische Energie. Mit r1 wird der Abstand der Teilchen zueinander ausgerechnet.
!abweichung gibt die gemittelte,betragliche Abweichung vom Grundzustand. ortsbetrag ist
!ein dummy für die Berechnung der Ausdehnung des Systems. rw,phiw und thetaw sind die Kugelko-
!ordinaten der Winkelgeschwindigkeit des Systems.
DOUBLE PRECISION, EXTERNAL :: kraft
!Subroutine zur berechnung der Ableitungen der Koordinaten
DOUBLE PRECISION, DIMENSION(6*N) :: y = 0.d0, r0=0.d0,r01=0.d0
!y beinhaltet die Koordinaten und Impulse.In r0 ist der Grundzustand zu finden.r01 ist dummy.
DOUBLE PRECISION, DIMENSION(3) :: schwerp = 0.d0,drehimp=0.d0,mitte=0.d0,winkelg,winkelg1=0.d0
!Mit schwerp wird der Schwerpunktsimpuls berechnet, mit mitte der Massenschwerpunkt. winkelg1
!ist für Zwischenrechnungen da.
REAL :: zeit1,zeit2
DOUBLE PRECISION, DIMENSION(3,3) :: drehz,drehy
!Drehmatrizen zur Rotation des Grundzustands
DOUBLE PRECISION, DIMENSION(3*N) :: l=0.d0

!Legen der Teilchen auf ein kubisches Gitter. Nur zu Beginn nötig gewesen.
!do i=0,5                !legt die Teilchen auf ein kubisches Gitter zwischen
!  do j=0,5              !den Koordinaten (0,0,0) und (25.8,25.8,25.8)
!    do k=0,5
!          y(6*k+1+6*6*j+6*6*6*i)=dble(k)*4.3d0
!      y(6*k+2+6*6*j+6*6*6*i)=dble(j)*4.3d0
!      y(6*k+3+6*6*j+6*6*6*i)=dble(i)*4.3d0
!    end do
!  end do
!end do

open(unit=7,file="grundzustand.txt",status="old",iostat=iostat,action="read")
if (iostat == 0) then
  do i=1,6*N
    read(7,*) y(i)
  end do
else
  write(*,*) 'es ist fehler',iostat,'aufgetreten,Grundzustand'
  stop
end if
close(unit=7,iostat=iostat)
write(*,*) iostat
r0 = y
kinetische = 0.d0
potentielle = 0.d0
energie = 0.d0
temperatur = 4700.d0
```

```
72  zeit = 0.d0
73  dt = 3.d-3                              !1.d-3 bei 8000K noch gut
74  zeitende = dt*20000.d0
75  anzahl = int((zeitende-zeit)/dt)
76  !Verteilen der Impulse
77  call montecarlo(temperatur,y)
78  do i=0,N-1
79    energie = energie + (y(6*i+4)*y(6*i+4)+y(6*i+5)*y(6*i+5)+y(6*i+6)*y(6*i+6))/(2.d0*masse)
80  end do
81  e0 = 3.0d0/2.0d0 *kb*temperatur
82  write(*,*) energie/(e0*N)
83  !Bestimmung Schwerpunktsimpuls
84  do i=0,N-1
85    schwerp(1) = schwerp(1) + y(6*i+4)
86    schwerp(2) = schwerp(2) + y(6*i+5)
87    schwerp(3) = schwerp(3) + y(6*i+6)
88  end do
89  schwerp = schwerp/N
90  write(*,*) "Schwerpunktsimpuls",schwerp
91  !Transformation ins Schwerpunktsystem
92  do i=0,N-1
93    y(6*i+4) = y(6*i+4) - schwerp(1)
94    y(6*i+5) = y(6*i+5) - schwerp(2)
95    y(6*i+6) = y(6*i+6) - schwerp(3)
96  end do
97  !Massenmittelpunkt
98  do i=0,N-1
99    mitte(1) = mitte(1) + y(6*i+1)
100   mitte(2) = mitte(2) + y(6*i+2)
101   mitte(3) = mitte(3) + y(6*i+3)
102 end do
103 mitte = mitte/N
104 !Verschieben des Schwerpunkts auf 0 ->Relativkoordinaten
105 do i=0,N-1
106   r0(6*i+1) = r0(6*i+1)-mitte(1)
107   r0(6*i+2) = r0(6*i+2)-mitte(2)
108   r0(6*i+3) = r0(6*i+3)-mitte(3)
109 end do
110 call drehimpuls(y,winkelg)
111 !Winkelgeschwindigkeit in Kugelkoordinaten
112 rw = Sqrt(winkelg(1)**2 +winkelg(2)**2 +winkelg(3)**2)
113 phiw = atan2(winkelg(2),winkelg(1))
114 thetaw = acos(winkelg(3)/rw)
115 !Ausgabe der Kugelkoordinaten der Winkelgeschwindigkeit und der Schrittweite
116 open(unit=33,file="winkel.txt",status="old",iostat=iostat,action="write")
117 if (iostat == 0) then
118   write(33,*) rw
119   write(33,*) phiw
120   write(33,*) thetaw
121   write(33,*) dt
122 else
123   write(*,*) 'es␣ist␣fehler',iostat,'aufgetreten,␣winkel.txt'
124   stop
125 end if
126 close(unit=33,iostat=iostat)
127
128 !Drehmatrizen
129 drehz=reshape((/cos(-phiw),sin(-phiw),0.d0,-sin(-phiw),cos(-phiw),0.d0,0.d0,0.d0,1.d0 /)&
130       &,(/3,3/))
131 drehy=reshape((/cos(-thetaw),0.d0,-sin(-thetaw),0.d0,1.d0,0.d0,sin(-thetaw),0.d0&
132       &,cos(-thetaw)/),(/3,3/))
133 !Test, ob Rotationen korrekt, wenn korrekt: so steht w in z-Richtung; x- u. y-Komponente = 0
134 do j=1,3
135   do k=1,3
136     winkelg1(j) = winkelg1(j)+drehz(j,k)*winkelg(k)
137   end do
138 end do
139 write(*,*) "winkelg1",winkelg1
140 winkelg = 0.d0
141 do j=1,3
142   do k=1,3
143     winkelg(j) = winkelg(j)+drehy(j,k)*winkelg1(k)
144   end do
```

```
145 end do
146 write(*,*) "Winkelgeschwindigkeit␣transformiert", winkelg
147 !Rotation des Grundzustandsvektors
148 do i=0,N-1
149   do j=1,3
150     do k=1,3
151       r01(6*i+j) = r01(6*i+j)+drehz(j,k)*r0(6*i+k)
152     end do
153   end do
154 end do
155 r0 = 0.d0
156 do i=0,N-1
157   do j=1,3
158     do k=1,3
159       r0(6*i+j) = r0(6*i+j)+drehy(j,k)*r01(6*i+k)
160     end do
161   end do
162 end do
163 !Rotation der Orts- und Impulskoordinaten von y
164 r01 = 0.d0
165 do i=0,N-1
166   do j=1,3
167     do k=1,3
168       r01(6*i+j) = r01(6*i+j)+drehz(j,k)*y(6*i+k)
169       r01(6*i+j+3) = r01(6*i+j+3)+drehz(j,k)*y(6*i+k+3)
170     end do
171   end do
172 end do
173 y = 0.d0
174 do i=0,N-1
175   do j=1,3
176     do k=1,3
177       y(6*i+j) = y(6*i+j)+drehy(j,k)*r01(6*i+k)
178       y(6*i+j+3) = y(6*i+j+3)+drehy(j,k)*r01(6*i+k+3)
179     end do
180   end do
181 end do
182
183 !l enthält die Kugelkoordinaten des gedrehten Grundzustands (r,phi,theta),
184 !sodass man jetzt, da w in r-Richtung steht, auf Phi nach jedem Iterationsschritt
185 !rw*dt aufaddieren kann, sich der Grundzustand also mitdreht.
186 do i=0,N-1
187   l(3*i+1) = sqrt(r0(6*i+1)**2 +r0(6*i+2)**2 +r0(6*i+3)**2)      !Abstand zu Mittelpunkt
188   l(3*i+2) = atan2(r0(6*i+2),r0(6*i+1))                          !Winkel Phi
189   l(3*i+3) = acos(r0(6*i+3)/l(3*i+1))                            !Winkel Theta
190 end do
191
192 call cpu_time(zeit1)
193 open(unit=44,file="drehimpuls.txt",status="old",iostat=iostat,action="write")
194 open(unit=8,file="test.txt",status="old",iostat=iostat,action="write")
195 open(unit=23,file="trajektorien.txt",status="old",iostat=iostat,action="write")
196 if (iostat == 0) then
197 do i=1, anzahl
198     zeit = dble(i-1)*dt
199     kinetische = 0.d0
200     potentielle = 0.d0
201     ort = 0.d0
202     nenner2 = 0.d0
203     abweichung = 0.d0
204     !Berechnung der potentiellen Energie
205     do j=0,N-2
206       do k=j+1,N-1
207         r1 = sqrt((y(6*j+1)-y(6*k+1))**2 +(y(6*j+2)-y(6*k+2))**2 +(y(6*j+3)-y(6*k+3))**2)
208         potentielle = potentielle + a*exp(-0.5d0*(r1/s1)**2)-b*exp(-0.5d0*(r1/s2)**2)
209       end do
210     end do
211     !Berechnung der Ausdehnung des Arrangements
212     do j=0,N-1
213       ort = ort + sqrt(y(6*j+1)**2 +y(6*j+2)**2 +y(6*j+3)**2)
214     end do
215     !Berechnung der kinetischen Energie
216     do j=0,N-1
217       kinetische = kinetische + (y(6*j+4)*y(6*j+4)+y(6*j+5)*y(6*j+5)+&
```

```
                    &y(6*j+6)*y(6*j+6))/(2.d0*masse)
    end do
    !Berechnung der Abweichung von dem Grundzustand
    do j=0,N-1
      ortsbetrag = Sqrt((y(6*j+1)-r0(6*j+1))**2 &
                   &+ (y(6*j+2)-r0(6*j+2))**2 +(y(6*j+3)-r0(6*j+3))**2)
      if (ortsbetrag < 15.d0) then
        abweichung = abweichung + ortsbetrag
        nenner2 = nenner2 + 1.d0
      end if
    end do
    write(8,'(1x,g11.6,6(1x,G22.15))') zeit,kinetische,potentielle&
                &,potentielle+kinetische,ort/n,abweichung/nenner2
    call rungekutta4(dt,zeit,y,kraft)
!**********************Korrektur des Drehimpulses*********************************************
    !Addition des durch die Winkelgeschwindigkeit veränderten Winkels in Phi
    do j=0,N-1
      l(3*j+2) = l(3*j+2) + rw*dt
    end do
    !Umschreiben in kartesische Koordinaten
    do j=0,N-1
      r0(6*j+1) = l(3*j+1)*sin(l(3*j+3))*cos(l(3*j+2))
      r0(6*j+2) = l(3*j+1)*sin(l(3*j+3))*sin(l(3*j+2))
      r0(6*j+3) = l(3*j+1)*cos(l(3*j+3))
    end do
!Nach 50 Iterationsschritten sollen die Trajektorien von 5 Teilchen ausgegeben werden
    if (modulo(i,50) == 0) then
      write(23,'(3(1x,G22.15))') y(99*6+1),y(99*6+2),y(99*6+3)
      write(23,'(3(1x,G22.15))') y(98*6+1),y(98*6+2),y(98*6+3)
      write(23,'(3(1x,G22.15))') y(97*6+1),y(97*6+2),y(97*6+3)
      write(23,'(3(1x,G22.15))') y(93*6+1),y(93*6+2),y(93*6+3)
      write(23,'(3(1x,G22.15))') y(92*6+1),y(92*6+2),y(92*6+3)
    end if
!Nach 200 Iterationsschritten soll der Zustandsvektor ausgegeben werden
    if (modulo(i,200)== 0) then
      open(unit=9,file="endzustand.txt",status="old",iostat=iostat,action="write")
      if (iostat == 0) then
        do j=1,6*N
          write(9,*) y(j)
        end do
      else
        write(*,*) 'es␣ist␣fehler',iostat,'aufgetreten,Zustandsvektor'
        stop
      end if
      close(unit=9,iostat=iostat)
      write(*,*) iostat
      schwerp = 0.d0
      do j=0,N-1
        schwerp(1) = schwerp(1) + y(6*j+4)
        schwerp(2) = schwerp(2) + y(6*j+5)
        schwerp(3) = schwerp(3) + y(6*j+6)
      end do
      write(*,*) 'Dies␣ist␣ein␣Test,␣ob␣der␣Schwerpunktsimpuls␣verschwunden␣ist'
      write(*,*) schwerp/N
      drehimp = 0.d0
      do j=0,N-1
        drehimp(1) = drehimp(1)+(y(6*j+2))*y(6*j+6)-(y(6*j+3))*y(6*j+5)
        drehimp(2) = drehimp(2)+(y(6*j+3))*y(6*J+4)-(y(6*j+1))*y(6*j+6)
        drehimp(3) = drehimp(3)+(y(6*j+1))*y(6*j+5)-(y(6*j+2))*y(6*j+4)
      end do
      write(*,*) "Drehimpuls",drehimp
      write(44,*) drehimp
    end if
!Ende der Ausgabe
  end do
else
  write(*,*) 'es␣ist␣fehler',iostat,'aufgetreten,test␣und␣trajektorie'
  stop
end if
close(unit=8,iostat=iostat)
write(*,*) iostat

open(unit=9,file="endzustand.txt",status="old",iostat=iostat,action="write")
```

```
291 if (iostat == 0) then
292   do i=1,6*N
293     write(9,*) y(i)
294   end do
295 else
296   write(*,*) 'es␣ist␣fehler',iostat,'aufgetreten,endzustand'
297   stop
298 end if
299 close(unit=9,iostat=iostat)
300 write(*,*) iostat
301
302 call cpu_time(zeit2)
303 write(*,*) zeit2-zeit1
304
305 END PROGRAM newtonrk4
306 !*****************************************************************************************
307 !Subroutine für die Integration der DGL
308 !In F müssen an der i-ten Stelle die Ableitungen von y(i) nach T stehen.Die Integration
309 !erfolgt über die Variable T. H ist die Schrittweite in T.
310 SUBROUTINE RUNGEKUTTA4(H,T,Y,F)
311 USE konstanten
312 IMPLICIT NONE
313 DOUBLE PRECISION, EXTERNAL :: F
314 DOUBLE PRECISION, DIMENSION(6*N), intent(inout) :: y
315 DOUBLE PRECISION, INTENT(INOUT)  :: T
316 DOUBLE PRECISION, INTENT(IN)     :: H
317 DOUBLE PRECISION, DIMENSION(6*N) :: k1,k2,k3,HF
318 DOUBLE PRECISION                 :: T1,hh,h6
319 INTEGER                          :: i
320 hh = h*0.5d0
321 h6 = h/6.0d0
322 call F(T,Y,k1)
323 T1 = T + hh
324 HF = y + hh*k1
325 call F(T1,HF,k2)
326 HF = y + hh*k2
327 call F(T1,HF,k3)
328 HF = y + k3*H
329 k2 = k2 + k3
330 T = T + H
331 call F(T,HF,k3)
332 y = y + h6 *(k1+2.0d0*k2+k3)
333 !Falls ein Teilchen nach einem Iterationsschritt in einer Dimension außerhalb des Intervalls
334 ![-25,25] liegt, so wird es an der anderen Seite wieder eingeführt. Die Ausgabe "sprung" zeigt
335 !während der Berechnungen an, ob ein Teilchen außerhalb des Kastens lag und kann mit etwaigen
336 !Sprüngen der Gesamtenergie identifiziert werden.
337 do i=0,N-1
338   if (y(6*i+1)>25.d0) then
339     y(6*i+1) = -25.d0
340     write(*,*) "sprung"
341   end if
342   if (y(6*i+2)>25.d0) then
343     y(6*i+2) = -25.d0
344     write(*,*) "sprung"
345   end if
346   if (y(6*i+3)>25.d0) then
347     y(6*i+3) = -25.d0
348     write(*,*) "sprung"
349   end if
350   if (y(6*i+1)<-25.d0) then
351     y(6*i+1) = 25.d0
352     write(*,*) "sprung"
353   end if
354   if (y(6*i+2)<-25.d0) then
355     y(6*i+2) = 25.d0
356     write(*,*) "sprung"
357   end if
358   if (y(6*i+3)<-25.d0) then
359     y(6*i+3) = 25.d0
360     write(*,*) "sprung"
361   end if
362 end do
363 RETURN
```

```
364 END SUBROUTINE RUNGEKUTTA4
365 !*****************************************************************************************
366 !Subroutine enthält die Bewegungsgleichungen/ die Ableitungen für den Vektor y(i)
367 SUBROUTINE KRAFT(T,Y,HF)
368 USE konstanten
369 IMPLICIT NONE
370 DOUBLE PRECISION, DIMENSION(6*N), INTENT(IN) :: Y
371 DOUBLE PRECISION, INTENT(IN) :: T
372 DOUBLE PRECISION, DIMENSION(6*N), INTENT(INOUT) :: HF
373 INTEGER :: i,j
374 DOUBLE PRECISION, PARAMETER :: a1 = a/(s1*s1),a2 = b/(s2*s2),a3=-0.5d0/(s1*s1)&
375                                &,a4=-0.5d0/(s2*s2)
376 DOUBLE PRECISION :: r, temp !r beinhaltet |r-r'|, temp wie im Abschnitt
377 HF = 0.d0                   !"Potential und effektive Wechselwirkung"
378 !Gleichung für dx/dt = p/m, Ableitungen der Ortskoordinaten
379 do i=0,N-1
380   HF(6*i+1) = y(6*i+4)/masse
381   HF(6*i+2) = y(6*i+5)/masse
382   HF(6*i+3) = y(6*i+6)/masse
383 end do
384 !Gleichung für dp/dt= F, Ableitungen der Impulskoordinaten
385 do i=0,(N-2)
386   do j=i+1,(N-1)
387     r = sqrt((y(6*i+1)-y(6*j+1))**2 + (y(6*i+2)-y(6*j+2))**2 + (y(6*i+3)-y(6*j+3))**2)
388     temp= a1*exp(a3*r*r)-a2 *exp(a4*(r)*(r))
389     HF(6*i+4) = HF(6*i+4) + temp*(y(6*i+1)-y(6*j+1))
390     HF(6*j+4) = HF(6*j+4) - temp*(y(6*i+1)-y(6*j+1))
391     HF(6*i+5) = HF(6*i+5) + temp*(y(6*i+2)-y(6*j+2))
392     HF(6*j+5) = HF(6*j+5) - temp*(y(6*i+2)-y(6*j+2))
393     HF(6*i+6) = HF(6*i+6) + temp*(y(6*i+3)-y(6*j+3))
394     HF(6*j+6) = HF(6*j+6) - temp*(y(6*i+3)-y(6*j+3))
395   end do
396 end do
397 !Reibungsterm: dp/dt = F "-gamma p". Nur am Anfang nötig gewesen
398 !do i=0,N-1
399 !  HF(6*i+4) = HF(6*i+4) - gamma*y(6*i+4)
400 !  HF(6*i+5) = HF(6*i+5) - gamma*y(6*i+5)
401 !  HF(6*i+6) = HF(6*i+6) - gamma*y(6*i+6)
402 !end do
403 RETURN
404 END SUBROUTINE KRAFT
405
406 !*****************************************************************************************
407 !Diese Subroutine dient der Impulsvergabe nach der Maxwell Boltzmann Verteilung mit Hilfe
408 !einer Monte Carlo Methode.
409 SUBROUTINE MONTECARLO(T,y)
410 USE konstanten
411 IMPLICIT NONE
412 DOUBLE PRECISION, INTENT(IN) :: T
413 DOUBLE PRECISION, DIMENSION(6*N), INTENT(INOUT) :: y
414 DOUBLE PRECISION        :: vert,test,theta,phi,betrag,random@,vertmax,betragmax
415 INTEGER :: i,j
416 !T ist die Temperatur, vert ist der zufällig Maxwell Boltzmann Verteilungswert, theta ist der
417 !zufällig erhaltene Winkel theta des Impulses, gleiches gilt für phi und betrag.
418 !vertmax gibt eine Funktion für das Maximum der Verteilung. Diese Funktion ist ein Fit der
419 !Verteilungsmaxima bei verschiedenen Temperaturen und wurde mit Mathematica durchgeführt.
420 !betragmax ist der Impulsbetrag bei dem die Verteilung einen Wert von 10^-5 annimmt.
421
422 vertmax = 181.69723638904276d0- 49036.1596250d0/(T*T) + &   !Gleichungen gelten nur bis 10000K
423         &12007.293973803038d0/T - 0.2326018089098843d0 *T +&
424         &0.000201434927180197 92d0 *T*T - 9.889627385975435d-8 *T*T*T + &
425         &2.8465004602038517d-11 *T*T*T*T - 4.898019128141887d-15 *T*T*T*T*T + &
426         &4.950284303906514d-19 *T*T*T*T*T*T - 2.7071514724946854d-23 *T*T*T*T*T*T*T + &
427         &6.176463667836648d-28 *T*T*T*T*T*T*T*T
428 betragmax = 0.0017642832638790238d0+ 0.0012577681693344063d0*Sqrt(T)+0.002d0
429 write(*,*) vertmax,betragmax
430 do i=0,N-1
431   j = 0
432 100 j = j+1
433   vert = random@()*vertmax      !random@() generiert eine Zufallszahl aus dem Intervall (0,1]
434   theta = pi*random@()
435   phi = 2.d0*pi*random@()
436   betrag = betragmax*random@()
```

```
437   test = betrag*betrag/(2.d0*masse*kb*T)
438   if (test > 397.d0) then       !Überprüfung, ob Argument in E-Funktion nicht zu negativ
439     goto 100                    !Exp(-397)=10^-173=0, also vert größer -> neue Zufallszahlen!
440   else if (vert > (2.d0*kb*T*masse*pi)**(-3/2) *4.d0*pi*betrag*betrag*exp(-test)) then
441     if (j > 2.d6) then
442       write(*,*) "Mehr als 2.d6 Überschreitungen!"
443       stop      !Falls über 2*10^6 Impulsbeträge eines einzigen Teilchens nicht der Verteilung
444     end if      !entsprechen, so wird das Programm abgebrochen. (Kann höher gesetzt werden)
445     goto 100
446   end if
447   y(6*i+4) = betrag*sin(theta)*cos(phi) !Überführung in kartesische Koordinaten
448   y(6*i+5) = betrag*sin(theta)*sin(phi)
449   y(6*i+6) = betrag*cos(theta)
450 end do
451 RETURN
452 END SUBROUTINE MONTECARLO
453 !*****************************************************************************************
454 !Diese Subroutine berechnet den Drehimpuls und anschließend über das Gaußsche Eliminations-
455 !verfahren die Winkelgeschwindigkeit die der Körper besitzt. Diese ist nötig um den Grund-
456 !zustand des Körpers entsprechend mitzudrehen, um die Abweichung von diesem auszurechnen,
457 !außerdem kann mit der bekannten Winkelgeschwindigkeit das Koordinatensystem zur Darstellung
458 !der Trajektorien der 5 Teilchen entsprechend mitgedreht werden.
459 SUBROUTINE Drehimpuls(y,winkelg)
460 USE konstanten
461 IMPLICIT NONE
462 DOUBLE PRECISION, DIMENSION(6*N),INTENT(INOUT) :: y
463 DOUBLE PRECISION, DIMENSION(3), INTENT(INOUT)  :: winkelg
464 DOUBLE PRECISION, DIMENSION(3)   :: mitte=0.d0, drehimp=0.d0,vektor=0.d0
465 DOUBLE PRECISION, DIMENSION(3,3) :: traeg=0.d0
466 DOUBLE PRECISION, DIMENSION(3,4) :: temp1,temp2
467 INTEGER :: i,j
468
469 winkelg = 0.d0
470 !Drehimpulskorrektur, zunächst Bestimmung des Schwerpunkts
471 do i=0,N-1
472   mitte(1) = mitte(1) + y(6*i+1)
473   mitte(2) = mitte(2) + y(6*i+2)
474   mitte(3) = mitte(3) + y(6*i+3)
475 end do
476 mitte = mitte/N
477 write(*,*) mitte
478 !Verschiebung des Schwerpunkts auf (0,0,0), die Ortsanteile von y sind nun die Relativkoord.
479 do i=0,N-1
480   y(6*i+1) = y(6*i+1)-mitte(1)
481   y(6*i+2) = y(6*i+2)-mitte(2)
482   y(6*i+3) = Y(6*i+3)-mitte(3)
483 end do
484 !Berechnung des Drehimpulses des zusammengesetzten Körpers in Bezug zum Schwerpunkt
485 do i=0,N-1
486   drehimp(1) = drehimp(1)+y(6*i+2)*y(6*i+6)-y(6*i+3)*y(6*i+5)
487   drehimp(2) = drehimp(2)+y(6*i+3)*y(6*i+4)-y(6*i+1)*y(6*i+6)
488   drehimp(3) = drehimp(3)+y(6*i+1)*y(6*i+5)-y(6*i+2)*y(6*i+4)
489 end do
490 write(*,*) drehimp
491 !Trägheitstensor
492 do i=0,N-1
493   traeg(1,1) = traeg(1,1) + y(6*i+2)**2 +y(6*i+3)**2
494   traeg(2,2) = traeg(2,2) + y(6*i+1)**2 +y(6*i+3)**2
495   traeg(3,3) = traeg(3,3) + y(6*i+1)**2 +y(6*i+2)**2
496   traeg(1,2) = traeg(2,1) - y(6*i+1)*y(6*i+2)
497   traeg(1,3) = traeg(1,3) - y(6*i+1)*y(6*i+3)
498   traeg(2,3) = traeg(2,3) - y(6*i+2)*y(6*i+3)
499 end do
500 traeg(2,1) = traeg(1,2)
501 traeg(3,1) = traeg(1,3)
502 traeg(3,2) = traeg(2,3)
503 traeg = masse*traeg
504 !Gaußsches Eliminationsverfahren
505 do i=1,3
506   do j=1,3
507     temp1(i,j) = traeg(i,j)     !Es wird zunächst das lineare Gleichungssystem ine ine neue
508     temp1(i,4) = drehimp(i)     !umgeschrieben I*w=L, I ist Matrixwert, w und L Vektor vertig
509   end do                        !Die erste Spalte des Trägheitstensors bildet die Koeffizien-
```

```
510 end do                          !ten von w_1 usw. die vierte Spalte der neuen Matrix bildet
511 temp2 = temp1                   !der Drehimpuls
512 do i=1,4
513   temp1(3,i) = temp2(3,i) - temp2(3,1)/temp2(2,1) *temp2(2,i)!Eliminierung der ersten
514   temp1(2,i) = temp2(2,i) - temp2(2,1)/temp2(1,1) *temp2(1,i)!Einträge in Zeile 2 und 3
515 end do
516 temp2 = temp1
517 do i=1,4
518   temp1(3,i) = temp2(3,i) - temp2(3,2)/temp2(2,2) *temp2(2,i)!Eliminierung des zweiten
519 end do                                                       !Eintrags in Zeile 2
520 !Lösung der Winkelgeschwindigkeit
521 winkelg(3) = temp1(3,4)/temp1(3,3)
522 winkelg(2) = (temp1(2,4)- winkelg(3)*temp1(2,3))/temp1(2,2)
523 winkelg(1) = (temp1(1,4)- winkelg(3)*temp1(1,3)- winkelg(2)*temp1(1,2))&
524              &/temp1(1,1)
525 write(*,*) "Drehimpuls",drehimp
526 !Test,ob mit errechneter Winkelgeschwindigkeit durch Multiplikation mit Trägheitstensor wieder
527 !der Bahndrehimpuls erhalten wird.
528 do i=1,3
529   do j=1,3
530       vektor(i) = vektor(i) + traeg(i,j)*winkelg(j)
531   end do
532 end do
533 write(*,*) "berechnet", vektor
534 write(*,*) winkelg
535 RETURN
536 END SUBROUTINE drehimpuls
```

3.2 Potential und effektive Wechselwirkung

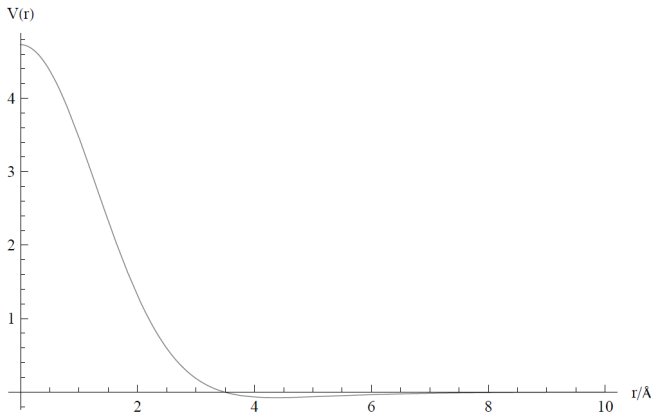
Als Ansatz für das Potential wurden zwei gaußsche Glockenkurven gewählt, wobei die eine ein positives und das andere ein negatives Vorzeichen besitzt. Die Summe der beiden Kurven ergibt das Potential. a und b geben die Höhen und σ_1 und σ_2 geben die Breiten der Glockenkurven an:

$$V(r) = a\,\mathrm{e}^{-\frac{1}{2}\left(\frac{r}{\sigma_1}\right)^2} - b\,\mathrm{e}^{-\frac{1}{2}\left(\frac{r}{\sigma_2}\right)^2} \tag{5}$$

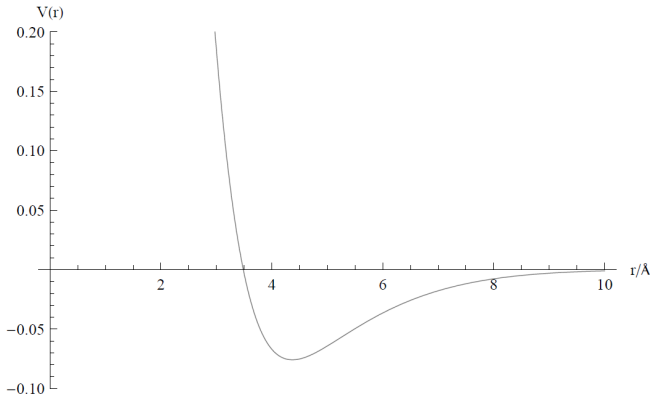
Geplant war ein System mit Teilchen der Masse von Silizium (28 u), einem Abstand zum Potentialminimum von etwa 4 Åund einer mittleren Bindungsenergie von 0.5 bis 1 eV. Durch mehrmaliges Erreichen des Grundzustands bei Änderung der Parameter, wurden als hinreichend annehmbare Parameter die folgenden gefunden.

$$a = 5.0\,\mathrm{eV} \qquad \sigma_1 = 1.3\,\text{Å} \qquad b = 0.27\,\mathrm{eV} \qquad \sigma_2 = 3.0\,\text{Å} \tag{6}$$

In Abbildung 2 sind zwei Plots des Potentials zu sehen. Das Minimum des Potentials liegt bei einem Abstand von 4.371 Å und besitzt eine Tiefe von -0.0759 eV.



(a) voller Wertebereich



(b) verkürzter Wertebereich

Abbildung 2: Plots des verwendeten Potentials.

Die Kraft, die von einem Teilchen auf das andere wirkt, errechnet sich aus dem negativen Gradienten des Potentials, wobei $r = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}$ zu setzen ist.

$$\vec{F}(\vec{r}_1, \vec{r}_2) = \underbrace{\left(\frac{a}{\sigma_1^2} \mathrm{e}^{-\frac{1}{2}\left(\frac{r}{\sigma_1}\right)^2} - \frac{b}{\sigma_2^2} \mathrm{e}^{-\frac{1}{2}\left(\frac{r}{\sigma_2}\right)^2} \right)}_{temp} \begin{pmatrix} x_1 - x_2 \\ y_1 - y_2 \\ z_1 - z_2 \end{pmatrix} \tag{7}$$

Dies ist die Kraft, die das Teilchen 2, welches sich an Ort $\vec{r}_2$ befindet, auf das Teilchen 1 an Ort $\vec{r}_1$ bewirkt. Die Berechnung dieser Kraft findet sich in der Subroutine `Kraft` wieder, welche in den Zeilen 384 - 396 zu finden ist. *temp* taucht als solches im Programm auf.
In der Subroutine `Kraft` werden die Ableitungen aller Koordinaten aus Gleichung 4 berechnet.

3.3 Gitter

Im nächsten Schritt wurden 216 Teilchen auf ein kubisches Gitter gesetzt. Entlang jeder Kantenlänge wurden 6 Teilchen gelegt und sie besaßen auf diesen einen Abstand von 4.3 Å. Siehe hierzu Abbildung 3. Die Umsetzung findet sich in den Zeilen 45 bis 54 in Abschnitt 3.1 wieder.

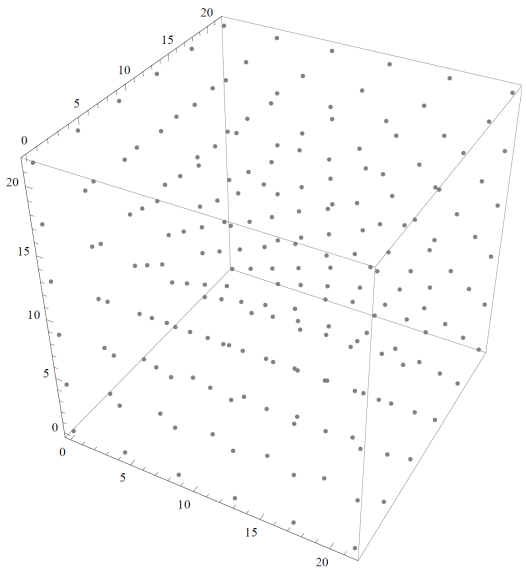


Abbildung 3: Anfangskomposition des Systems.

3.4 Grundzustand

Um in den Grundzustand zu gelangen, wurde der auskommentierte Reibungsterm in der Subroutine `Kraft` verwendet, Zeilen 397 - 402. Der Parameter *gamma* betrug den Wert $10\,\mathrm{ps}^{-1}$. Bei einer Schrittweite von $10^{-2}\,\mathrm{ps}$ und 20000 bis 30000 Iterationsschritten wurde der Grundzustand erreicht, siehe Abbildung 4. Die Verteilung wurde in grundzustand.txt abgespeichert und initialisiert am Anfang des Programms den Vektor y.

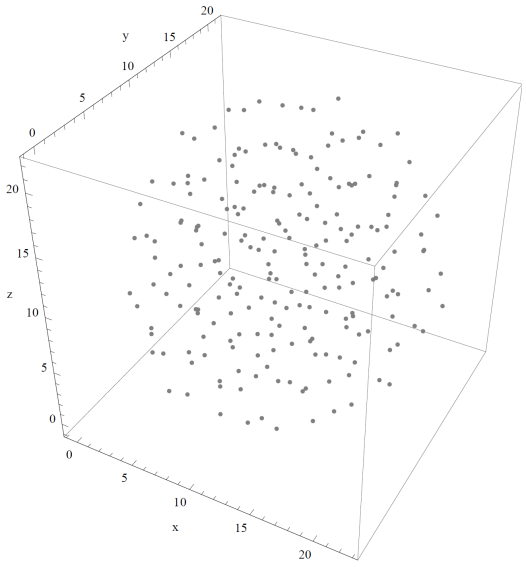


Abbildung 4: Grundzustand.

Zu bemerken ist, dass die Verteilung bei der geringen Teilchenzahl von 216 fast kugelsymmetrisch ist. Bei einer höheren Teilchenzahl, z.B. 1000, ist die anfängliche Gitterstruktur noch zu erkennen. Die wichtigsten Parameter des Grundzustands sind in der folgenden Tabelle festgehalten.

Tabelle 1: Signifikante Werte des Grundzustands.

$\langle\lvert\vec{r}_1 - \vec{r}_2\rvert\rangle$	Ausdehnung	Potentielle Energie	mittlere Bindungsenergie
4.681 Å	12.681 Å	-135.633 eV	0.628 eV

Die Bestimmung des mittleren Abstands zum nächsten Nachbarn $\langle\lvert\vec{r}_1 - \vec{r}_2\rvert\rangle$ ist abhängig vom Cut im Abstand. Der geringste Abstand liegt bei 3.135 Å, weshalb ein Cut von 6.270 Å zur Bestimmung des Grundzustandsabstands gewählt wurde. Anhand der Bindungsenergie kann bereits jetzt die Aussage getroffen werden ab welcher Temperatur das System, bei Annahme eines idealen Gases, vollständig gasförmig ist.

$$T = \frac{2 \cdot 0.628\,\mathrm{eV} \cdot \mathrm{K}}{3 \cdot 8.617 \cdot 10^{-5}\,\mathrm{eV}} = 4858\,\mathrm{K}$$

3.5 Impulsverteilung

Die Impulse wurden mit der Subroutine `montecarlo` in Abhängigkeit von der Temperatur verteilt, Zeilen 407 - 452. In der Subroutine werden per Zufallszahl ein Funktionswert und ein Impulsbetragswert, sowie die Winkel θ und ϕ generiert. Die verwendete Verteilung ist die Maxwell-Boltzmann-Verteilung:

$$\mathbf{p}(p) = 4\pi \left(2\pi\, k_b\, T\, m\right)^{-\frac{3}{2}} p^2 \exp\left(-\frac{p^2}{2m\, k_b\, T}\right) \tag{8}$$

Wenn die generierte Koordinate innerhalb der Verteilung liegt, wird der Impuls in kartesischen Koordinaten für das Teilchen abgespeichert. Sollte die Koordinate außerhalb der Verteilung liegen, so beginnt das Vorgehen von Neuem. Für jedes Teilchen wird dies bis zu $2{\cdot}10^6$ mal durchgeführt. Es wird eine Fehlermeldung ausgegeben und das Programm beendet, sobald diese Zahl überschritten wird.

3.6 Schwerpunktsystem und Drehimpuls

Nachdem dem System die Temperatur/ Energie zugeführt wurde, muss noch in das Schwerpunktsystem transformiert und mit dem Drehimpuls die Winkelgeschwindigkeit bestimmt werden. Dies ist wichtig für die Bestimmung des Beginns der Flüssigkeitsphase.
Der Schwerpunktimpuls wird in den Zeilen 83-90 nach der Formel

$$\vec{P}_S = \frac{\sum_i m_i \, \vec{p}_i}{\sum_i m_i} \tag{9}$$

errechnet. Anschließend wird durch eine Galilei-Transformation in das Schwerpunktsystem übergegangen, siehe hierzu Zeilen 91 - 96.
Für die Berechnung der Winkelgeschwindigkeit wird zunächst der Schwerpunkt in *mitte* ausgerechnet und dieser für die Vektoren y und $r0$ auf den Koordinatenursprung gelegt, sodass diese nun den Relativkoordinaten entsprechen. Die Winkelgeschwindigkeit wird in der Subroutine `drehimpuls`, Zeilen 454 - 536, berechnet. Nach Bestimmung des Drehimpulses und des Trägheitstensors des Systems wird über das Gaußsche Eliminationsverfahren die Winkelgeschwindigkeit errechnet.
Die Vektoren y und $r0$ werden im Anschluss so gedreht, dass die Ausrichtung der Winkelgeschwindigkeit entlang der z-Achse liegt. Nach jedem Aufruf von `rungekutta` wird der Grundzustandsvektor $r0$ um den Winkel $|\vec{\omega}| \cdot dt$ gedreht, damit die Abweichung zum Grundzustand bestimmt werden kann.

4 Ergebnisse

4.1 Parameter für die Aggregatzustände

Die Berechnungen wurden je nach Temperatur mit unterschiedlichen Schrittweiten, jedoch immer mit 20000 Schritten durchgeführt. Die untersuchten Parameter waren zum Einen der Plasmaparameter:

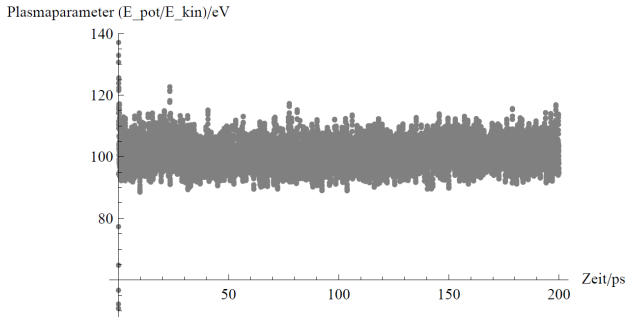
$$\eta = \frac{|E_{pot}|}{E_{kin}} \tag{10}$$

Wenn der Plasmaparameter η kleinergleich 1 ist, so ist das System vollständig gasförmig, da dann die kinetische Energie größer als die mittlere Bindungsenergie des Systems ist.
Zum anderen wurde die Ausdehnung des Systems betrachtet. In der Ausdehnung sind Oszillationen zu beobachten, die im festen Fall nicht abklingen. Beim Übergang von fest zu flüssig ist eine starke Dämpfung der Oszillation zu beobachten.
Die zur Charakterisierung des Aggregatzustands letzten Parameter waren die Abweichung vom Grundzustand und ausgewählte Trajektorien nebeneinander liegender Teilchen. Diese Parameter sind wichtig für den Übergang von fest zu flüssig. Das System ist flüssig, wenn die Abweichung vom Grundzustand größer als der Grundzustandsabstand ist oder wenn Teilchen die Gitterplätze verlassen und durch das Arrangement wandern können.
Bei allen Berechnungen wurde darauf geachtet, dass sich die Gesamtenergie über den beobachteten Zeitraum nicht mehr als ein halbes Prozent verändert hat. Bei einer Schrittweite $dt = 10^{-3}\,\mathrm{ps}$ wird diese Bedingung bis zu 8000 K erfüllt und bei Temperaturen unter 500 K genügt $dt = 10^{-2}\,\mathrm{ps}$.

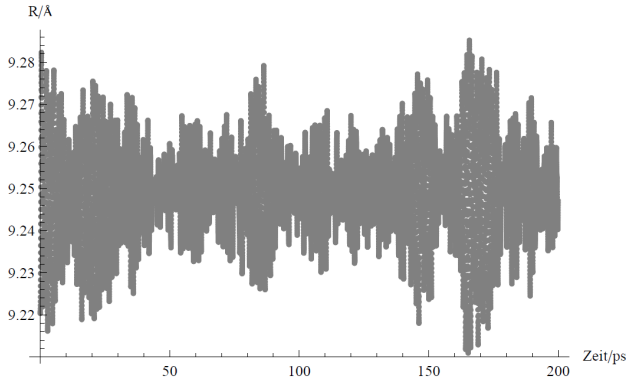
4.2 Aggregatzustände

4.2.1 Fest

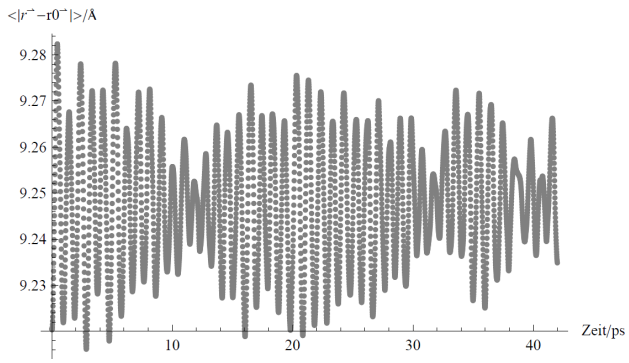
Typische Bilder des festen Zustands werden bei beispielsweise 100 K erhalten:



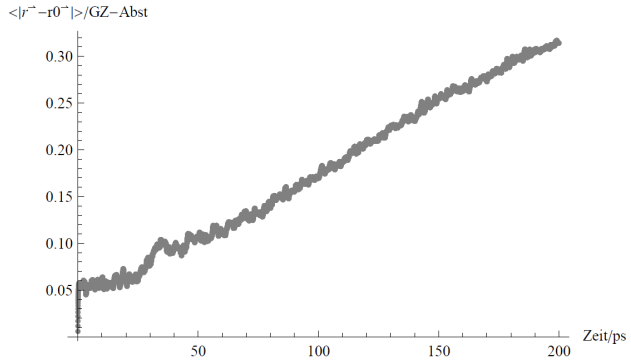
(a) Plasmaparameter.



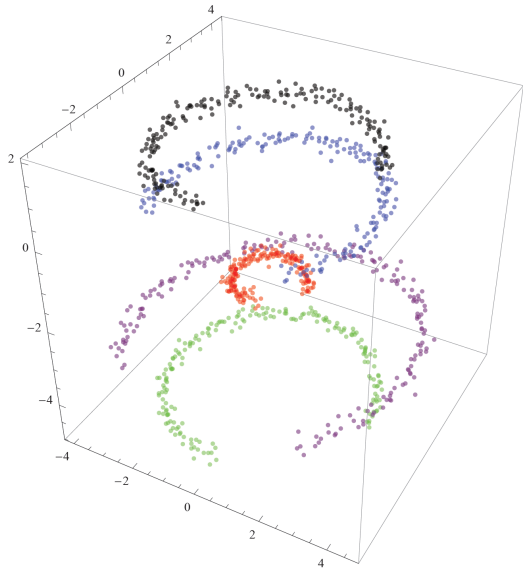
(b) Ausdehnung des Systems.



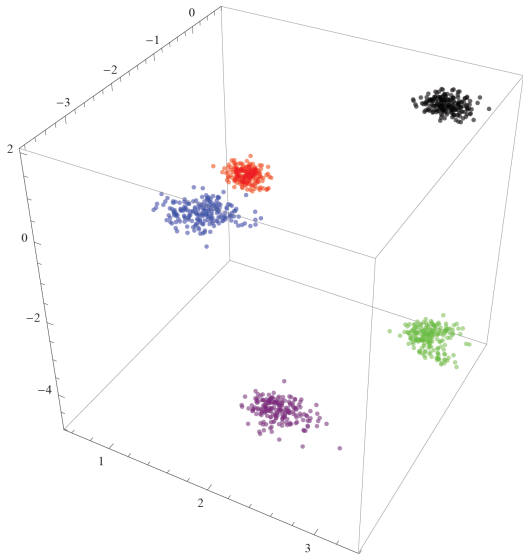
(c) Ausdehnung, kleinerer Ausschnitt.



(d) Abweichung von dem Grundzustand in Einheiten des Grundzustandsabstandes.



(e) Trajektorien, nicht mitrotiert.



(f) mitrotiertes Koordinatensystem.

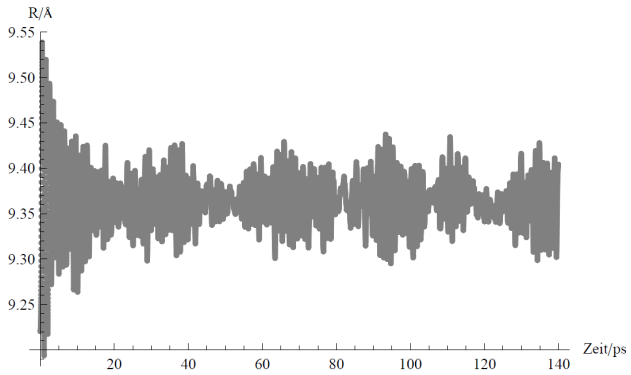
Abbildung 5: Plots der Parameter und der Trajektorien bei 100 K.

Es ist bei dem Plasmaparameter im festen Zustand nichts auffälliges zu erkennen. Bei der Ausdehnung des Systems sind Oszillationen mit einer Einhüllenden zu beobachten. Die Amplitude der Oszillation

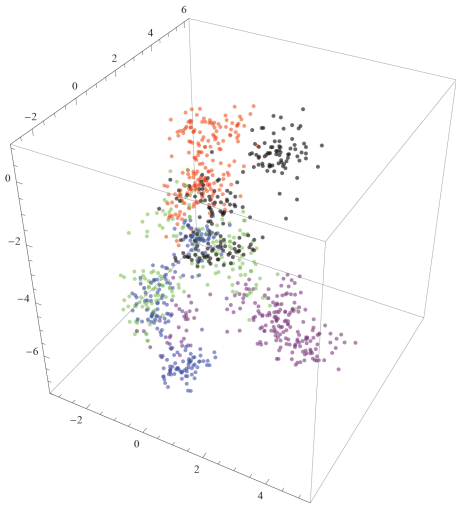
ist über den beobachteten Zeitraum näherungsweise kontant. In Abbildung 5(e) ist zu sehen, wie das Arrangement als ganzes rotiert. In der darauf folgenden Unterabbildung wurde das Koordinatensystem mit dem Arrangement rotiert. In der Abweichung vom Grundzustand, dass die Teilchen selbst bei 100 K in der Lage sind sich nach einer gewissen Zeit vom Gitterplatz zu bewegen, weshalb dieser Parameter im Folgenden nicht mehr berücksichtigt wird. Die Abweichung vom Grundzustand steigt mit der Zeit stetig an, woraus geschlossen werden kann, dass die Teilchen schon bei 100 K im gewissen Maß in der Lage sind sich von ihren Gitterplätzen zu entfernen.

4.2.2 Flüssig

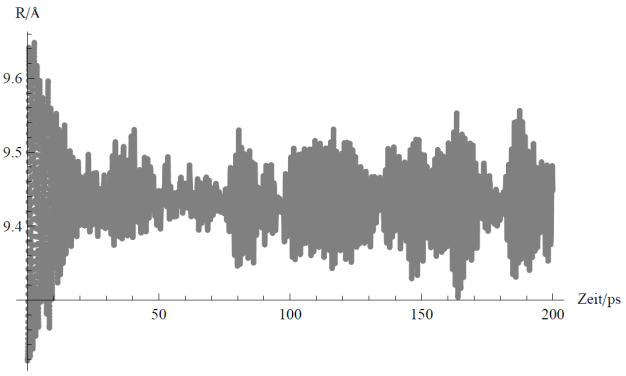
Zur Definition einer Flüssigkeit gehört, dass sich die Teilchen innerhalb des Körpers, hier Flüssigkeitstropfen, frei bewegen können. Aus diesem Grund spielen vor allem die ausgewählten Trajektorien eine entscheidende Rolle zur Charakterisierung der Flüssigkeitsphase, da die Abweichung von dem Grundzustand bereits im festen Zustand beliebig mit der Zeit ansteigt. Ein weiterer Indikator für den Übergang zur Flüssigkeit ist, wie sich zeigt, die Oszillation der Ausdehnung.
Das System besitzt keine konkrete Siedetemperatur, es kann nur ein grober Bereich des Übergangs angegeben werden. Die Abbildung 6 verdeutlicht dies. Auffällig ist, dass bei zunehmender Temperatur die Oszillation stärker gedämpft wird. Spätestens bei 1000 K kann das System als flüssig klassifiziert werden, da sich hier die Teilchen, in einem kleinen Zeitraum, deutlich von ihren ehemaligen Gitterplätzen entfernen sowie eine starke Dämpfung der Oszillation der Ausdehnung auftritt.
Bei weiterer Erhöhung der Temperatur erhöht sich auch die Beweglichkeit der Teilchen und es werden Bilder wie in Abbildung 7 erhalten. Die Teilchen bewegen sich bereits bei 2000 K, wie an den Trajektorien zu sehen ist, innerhalb des “Tropfens” vollkommen frei. Bei 3000 K werden die ersten Teilchen abgedampft, da deren kinetische Energie größer als die Bindungsenergie ist. Die Sprünge in der Gesamtenergie hängen damit zusammen, dass Teilchen die nach Anwendung von `rungekutta4` außerhalb des Kastens mit den Kantenlängen [-25,25] liegen an der anderen Seite wieder eingeführt werden.



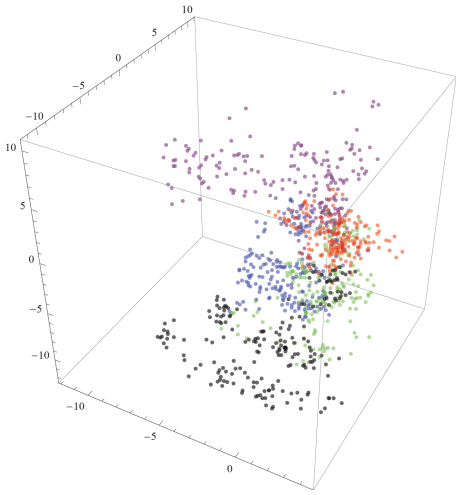
(a) Ausdehnung bei 500 K.



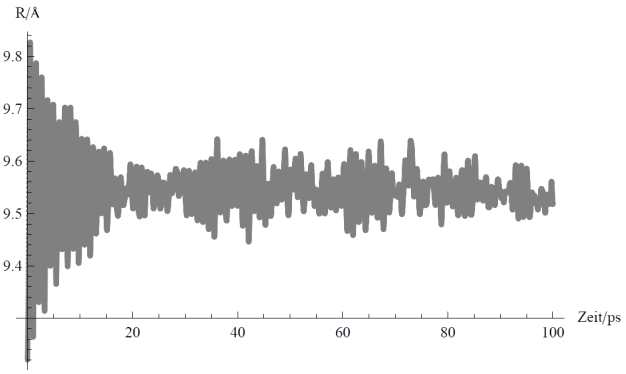
(b) Trajektorien bei 500 K.



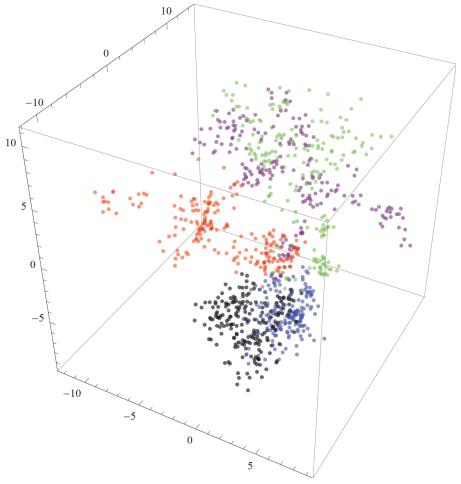
(c) Ausdehnung bei 700 K.



(d) Trajektorien bei 700 K.

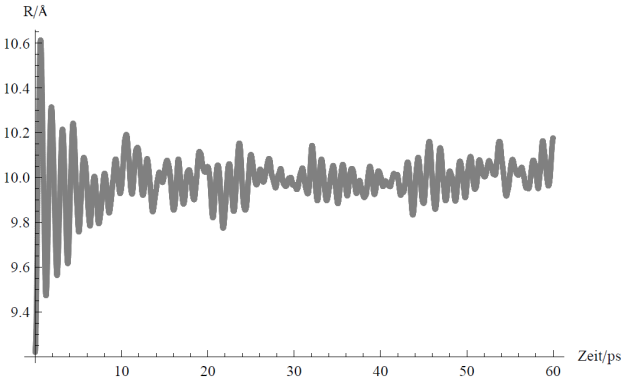


(e) Ausdehnung bei 1000 K.

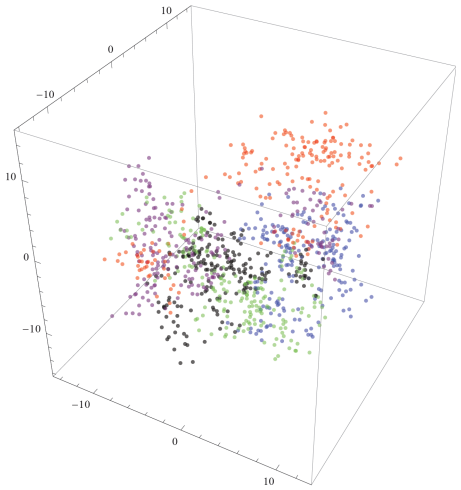


(f) Trajektorien bei 1000 K.

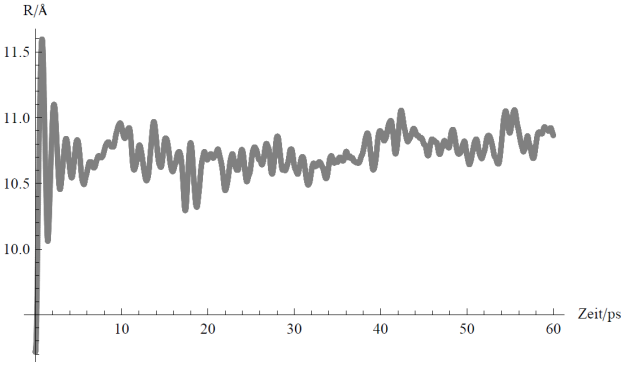
Abbildung 6: Übergang von fest zu flüssig.



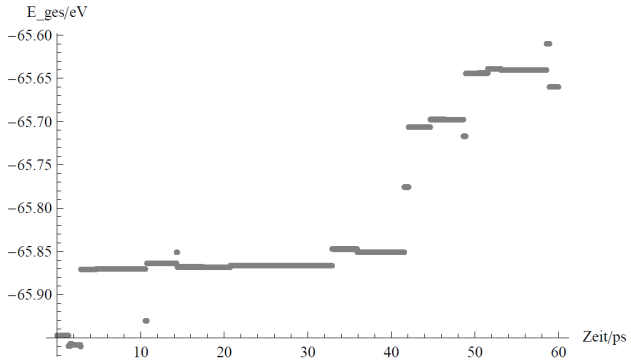
(a) Ausdehnung bei 2000 K.



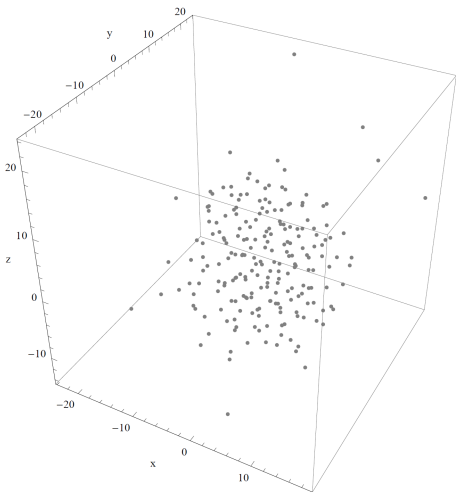
(b) Trajektorien bei 2000 K.



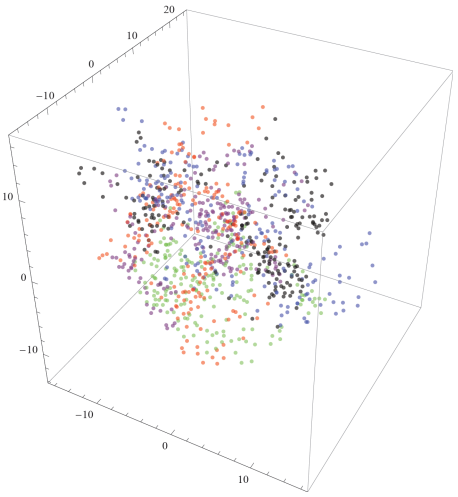
(c) Ausdehnung bei 3000 K.



(d) Gesamtenergie bei 3000 K.



(e) Endzustand bei 3000 K.

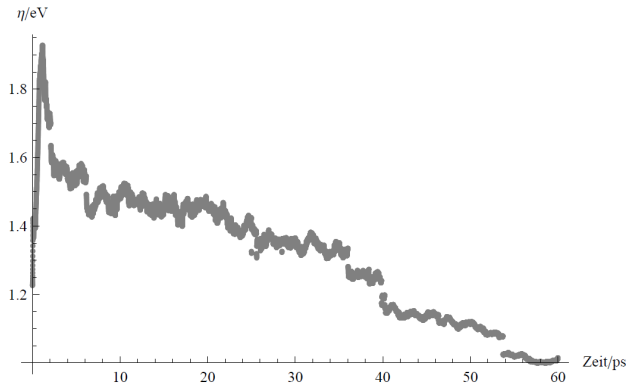


(f) Trajektorien bei 3000 K.

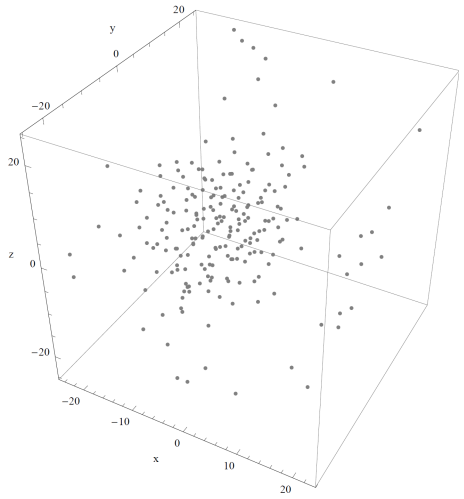
Abbildung 7: Plots der Flüssigkeitsphase.

4.2.3 Gasförmig

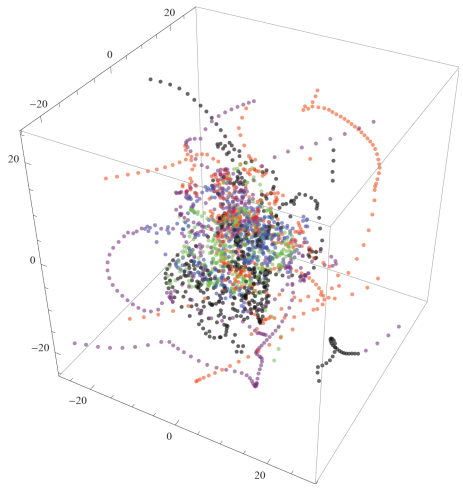
Charakteristisch für die Gasphase ist, wie bereits erwähnt, das Verhältnis von potentieller zu kinetischer Energie, der Plasmaparameter. Sobald dieser kleiner 1 ist, ist das System vollständig gasförmig. In Abbildung 8 sind Plots des Plasmaparameters, des Endzustands und der Trajektorien im Phasenübergang gegeben. Die Siedetemperatur befindet sich zwischen 4700 und 5000 K, da der Plasmaparameter bei 5000 K unter 1 fällt. Das System besitzt somit eine Siedetemperatur ähnlich einem idealen Gas. Auch wenn die mittlere kinetische Energie größer als die mittlere Bindungsenergie des Systems ist, liegen nicht alle Teilchen frei vor. Es bilden sich Cluster. Der größte stammt noch von der Anfangskonstellation.
In Abbildung 9 ist das System bei weiterer Energiezugabe zu sehen.



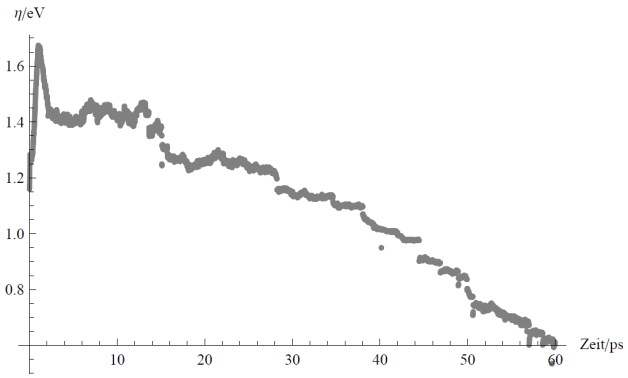
(a) Plasmaparameter bei 4700 K.



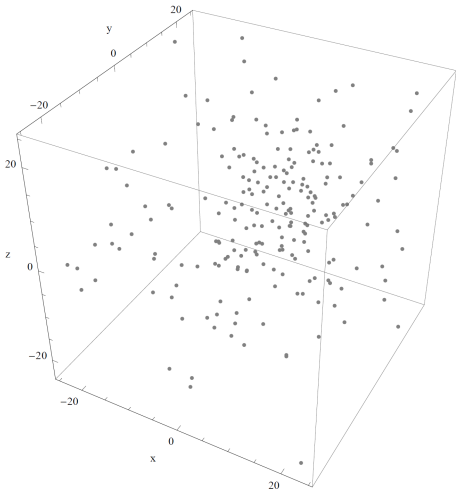
(b) Endzustand bei 4700 K.



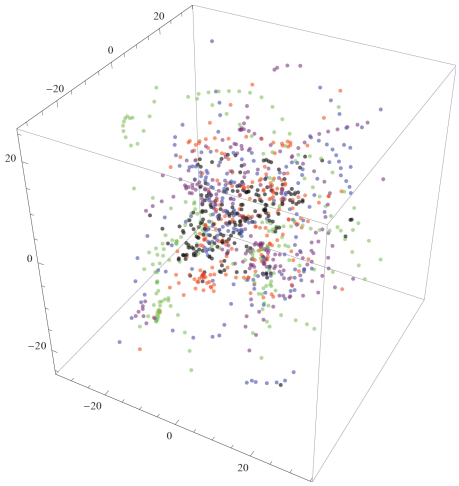
(c) Trajektorien bei 4700 K.



(d) Plasmaparameter bei 5000 K.

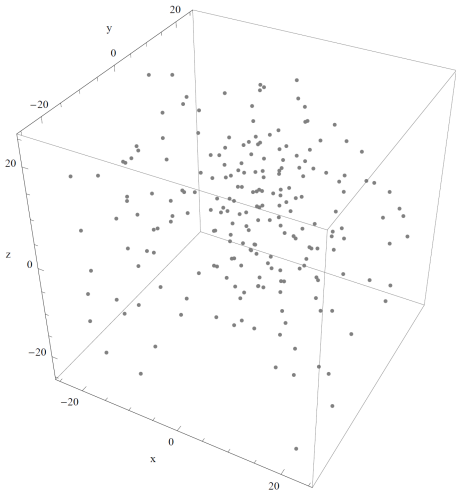


(e) Endzustand bei 5000 K.

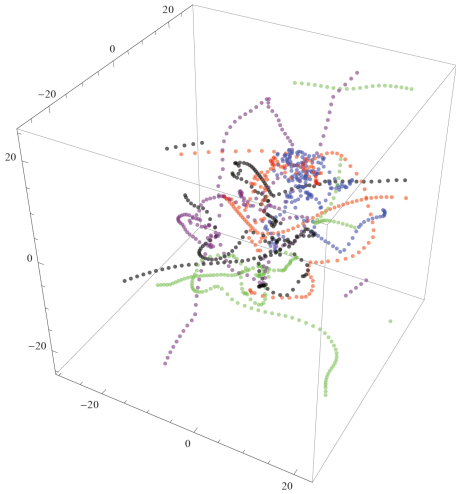


(f) Trajektorien bei 5000 K.

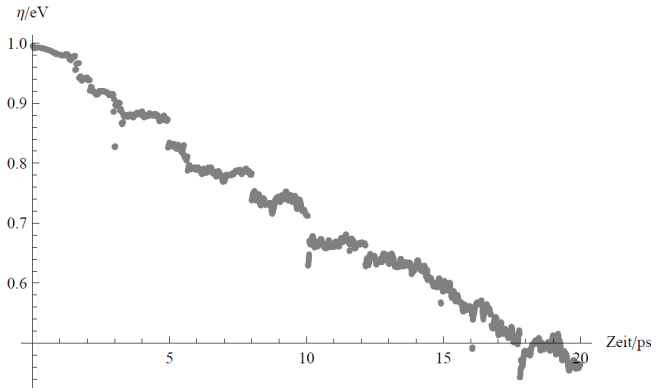
Abbildung 8: Übergang flüssig-gasförmig.



(a) Endzustand bei 6000 K.



(b) Trajektorien bei 6000 K.



(c) Plasmaparameter bei 6000 K.

Abbildung 9: Plots der Gasphase.

4.3 Zusammenfassung

Eine kurze Zusammenfassung der Ergebnisse findet sich in der folgenden Tabelle wieder.

Tabelle 2: Zusammenfassung der Phasenübergänge

Phasenübergang	Temperatur/K	Genutzte Parameter
fest zu flüssig	500 bis 1000	Ausdehnung und Trajektorien
flüssig zu gasförmig	4700 bis 5000	Plasmaparameter, Trajektorien und Endzustand