

Laskennallisen tekniikan erikoistyö
S-114.215

NEB-metodin toteutus MD-ohjelmaan Boundary2d

Tapio Nieminen
tapio.nieminen@hut.fi
51114E
23.08.2002

Sisältö

1	Esipuhe	2
2	NEB-metodi	3
3	NEB:n toteutus ja käyttö Boundary2d-ohjelmassa	5
3.1	Toteutus	5
3.2	Käyttöesimerkki	5
3.3	Säädinkohtaiset käyttöohjeet	6
4	Testiajojen tuloksia ja havaittuja ongelmia	9
4.1	Yleistä	9
4.2	Dislokaation liikkumisesimerkki	10
5	Johtopäätöksiä	12
A	Using NEB	12
B	Short Descriptions of NEB Functions and Procedures	15

1 Esipuhe

Tarkoituksena oli lisätä Boundary2d-nimiseen [3] MD-ohjelmaan koodi, jolla voidaan laskea minimienergiapolku kahden annetun tilan, alku- ja lopputilan, välillä. Tällainen minimienergiapolun ja samalla polun korkeimman kohdan eli satulapisteen löytäminen on tavallinen ja tärkeä tehtävä mm. fysikaalisessa kemiassa ja kiinteän aineen fysiikassa, sillä se määrää reaktionopeuden siirryttäessä alku-tilasta lopputilaan tai takaisin. Polusta nähdään myös, onko reaktio ”suoraviivainen” vai onko sillä kenties stabiileja välitiloja tai muita erikoisuuksia.

Eräs hyväksi havaittu algoritmi tämän ongelman ratkaisemiseksi on NEB (nudged elastic band), jonka Hannes Jónsson työryhmineen on kehittänyt Washingtonin yliopistossa. Satulapisteen eli kohdan, joka on jonkun funktion maksimi yhden koordinaatin suhteen ja minimi muiden koordinaattien suhteen, löytämiseksi on olemassa paljon muitakin metodeja, joista jotkut toimivat hyvin, jotkut toimivat huonosti ja jotkut toimivat hyvin, mutta vaativat esim. paljon laskentatehoa. NEB:n hyvänä puolena verrattaessa useimpiin näistä metodeista on se, että NEB löytää paitsi satulapisteen myös koko minimienergiapolun. Tosin kaikki muutkin ns. ”chain-of-states”-metodit tekevät tämän, mutta NEB on eräs hyvä tämän metodiryhmän edustaja. Tämän takia NEB-metodi valittiin Boundary-ohjelmaan lisättävän toiminnallisuuden pohjaksi.

Lopuksi tarkoituksena oli testata laajennettua Boundary-ohjelmaa muutamilla testitapauksilla ja katsoa, mitä tuloksista voidaan nähdä.

2 NEB-metodi

NEB-metodia on maailmalla käytetty useissa paikoissa ja siitä on olemassa useita variaatioita. Tähän tehtävään valittiin NEB-metodin kuvaus Graeme Henkelmanin väitöskirjasta [1], joka on tehty Jónssonin sijaintipaikassa Washingtonin yliopistossa ja kuvaa uusimman, parannellun version NEB-metodista. Myös vanhempaa Jónssonin ryhmän julkaisua [2] käytettiin hyväksi NEB:iä toteutettaessa.

NEB:n perusidea on luoda tunnettujen alku- ja lopputilan välillä joukko kuvia, jotka on yhdistetty toisiinsa jousivoimilla. Välikuvat voidaan luoda esim. lineaarisella interpolaatiolla, kuten tässä tapauksessa on tehty. Alku- ja loppukuvat pidetään kiinnitettyinä NEB-algoritmissa eli niiden atomit eivät liiku. Välikuvat saavat vapaasti relaksoitua polkua eli moniulotteista käyrää, joka yhdistää kuvat toisiinsa, kohtisuorassa olevissa suunnissa. Polun suunnassa ja ainoastaan siinä suunnassa vaikuttaa jousivoima, jonka tehtävänä on pitää kuvien etäisyys vakiona. Polun suunnassa ei vaikuta muita voimia kuin jousivoima. Tällä periaatteella kuvien pitäisi liikkua kohti minimienergiapolkua todellisten atomien välisten voimien polkua vastaan kohtisuorien komponenttien vaikutuksesta. Kun tätä diskretoitua systeemiä sitten iteroidaan eteenpäin tietokoneella, tarpeeksi monen aika-askeleen kuluttua kuvat ajautuvat riittävällä tarkkuudella minimienergiapolulle. Tässä metodissa ei kuitenkaan ole varsinaista satulapisteen löytöominaisuutta; mikään kuva ei varsinaisesti suppene kohti satulapistettä vaan kaikki kuvat suppenevat kohti minimienergiapolkua ja nolasta muutamaa kuvaa tulee näin suhteellisen lähelle satulapistettä kuvien määrästä ja tilanteesta riippuen. Varsinainen satulapiste voidaan tarkemmin määrittää muilla metodeilla tai interpoloimalla NEB:n antamien tulospisteiden pohjalta.

Jónssonin ryhmä on kehittänyt NEB:stä useita versioita, jotka on pääpiirteissään esitellyt G. Henkelmanin väitöskirjassa [1]. Näissä eri versioissa suurimpana erona on polun suunnan eli tangentin estimaatin laskeminen. G. Henkelmanin väitöskirjassa [1] esitetyssä NEB:n uusimmassa versiossa myös jousivoimia on hieman muutettu.

Tämän uusimman NEB:n version tangentti τ kuvan i kohdalla on:

$$\tau_i = \begin{cases} \tau_i^+ & \text{jos } V_{i+1} > V_i > V_{i-1} \\ \tau_i^- & \text{jos } V_{i+1} < V_i < V_{i-1} \end{cases}, \quad (1)$$

missä

$$\tau_i^+ = \mathbf{R}_{i+1} - \mathbf{R}_i \quad \text{ja} \quad \tau_i^- = \mathbf{R}_i - \mathbf{R}_{i-1} \quad (2)$$

ja \mathbf{R} on koordinaattivektori ja V potentiaalienergia. Molempien viereisten kuvien ollessa kuvaa i korkeammalla tai alempana potentiaalienergiassa kyseessä on ääriarvo. Tämän tangentti lasketaan näin:

$$\tau_i = \begin{cases} \tau_i^+ \Delta V_i^{max} + \tau_i^- \Delta V_i^{min} & \text{jos } V_{i+1} > V_i \\ \tau_i^+ \Delta V_i^{min} + \tau_i^- \Delta V_i^{max} & \text{jos } V_{i+1} < V_i \end{cases}, \quad (3)$$

missä

$$\begin{aligned} \Delta V_i^{max} &= \max(|V_{i+1} - V_i|, |V_{i-1} - V_i|) \\ \Delta V_i^{min} &= \min(|V_{i+1} - V_i|, |V_{i-1} - V_i|) \end{aligned} \quad (4)$$

NEB:ssä tarvittava jousivoiman tangentin suuntainen komponentti saadaan yhtälöstä

$$\mathbf{F}_i^s|_{||} = k(|\mathbf{R}_{i+1} - \mathbf{R}_i| - |\mathbf{R}_i - \mathbf{R}_{i-1}|)\hat{\tau}_i \quad (5)$$

Kun nämä on saatu laskettua, voidaan laskea kuvaan i vaikuttava kokonaisvoima

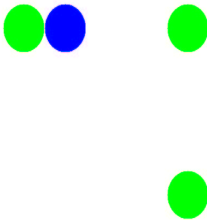
$$\mathbf{F}_i = \mathbf{F}_i^s|_{||} - \nabla V(\mathbf{R}_i)|_{\perp}, \quad (6)$$

missä ”todellisen” voiman vastaluku on potentiaalin gradientti

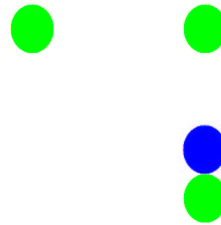
$$\nabla V(\mathbf{R}_i)|_{\perp} = \nabla V(\mathbf{R}_i) - \nabla V(\mathbf{R}_i) \cdot \hat{\tau}_i \hat{\tau}_i. \quad (7)$$

Tämän jälkeen voidaan kuvan i uudet koordinaatit saada laskemalla uusi nopeusvektori ja sen jälkeen uusi paikkavektori.

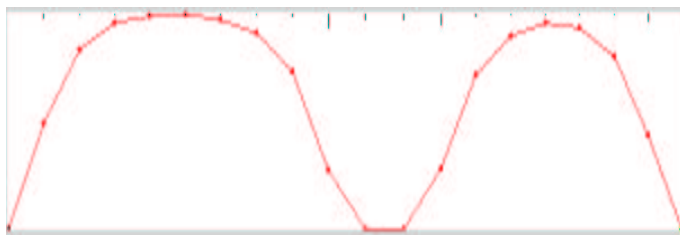
Seuraava esimerkki selvittää asiaa. Kuvassa 1 näkyy neljä atomia, joista vihreät ovat liikkumattomia. Sininen atomi lähtee liikkeelle ja liikkuu toisen vihreän atomin kautta kuvan 2 tilanteeseen. Sinisen atomin rata on polku ja siinä on kaksi satulapistettä potentiaalienergian suhteen. Nämä ovat ensimmäisen ja toisen sekä toisen ja kolmannen vihreän atomin välissä (kuvan 3 huiput). Näissä pisteissä koko systeemin (eli kaikki neljä atomia) potentiaalienergia on maksimissa polun suuntaisesti, mutta jos sininen atomi läh-
tisikin liikkumaan pois polulta (ylös tai alas ensimmäisestä välistä, oikealle tai vasemmalle toisesta välistä), potentiaalienergia kasvaisi lisää. Tällaisia ovat satulapisteet. Kuvassa 3 näkyy systeemin potentiaalienergia polun funktiona. ”Polku” tarkoittaa tässä yhteydessä ja myöhemminkin potentiaalienergian piirtämisestä polun suhteen puhuttaessa oikeasta polusta tehtyä yksiulotteista pituus- eli matkamittausta. Tässä esimerkissä polun koko pituus on sinisen atomin kulkema kokonaismatka ja paikka polulla on sinisen atomin siihen mennessä kulkema matka alkupisteestä laskien



Kuva 1. Alkutilanne



Kuva 2. Lopputilanne



Kuva 3: Potentiaalienergia polun funktiona

3 NEB:n toteutus ja käyttö Boundary2d-ohjelmassa

3.1 Toteutus

Tässä työssä pohjana käytettyä Boundaryn versiota on alkuperäisen julkaisun [3] jälkeen kehitetty monipuolisemmaksi ja tämä versio sekä sen manuaali on saatavilla lähteestä [4].

Edellä esitettyjen kaavojen perusteella NEB:n implementointi Boundaryyn oli suhteellisen suoraviivaista. Kaikki atomien tietorakenteet piti allokoida tarpeeksi suuriksi, jotta kuvat mahtuisivat niihin ja lisäksi piti tietenkin toteuttaa em. kaavojen mukainen iteraatiomekanismi. Olemassaolevaa Lennard-Jones -potentiaalin mukaista voimienlaskurutiinia voitiin käyttää pienin muutoksin.

Jotta NEB:tä varsinaisesti voisi käyttää, piti ohjelmaan lisätä alkutilojen luontimahdollisuus ja potentiaalienergian piirto polun funktiona. Ensimmäinen ja viimeinen kuva joko luodaan Boundarylla normaalisti tai luodaan tiedostosta ja välikuvat interpoloidaan tai luetaan tiedostosta. Kaikkia kuvia voidaan myös vapaasti muokata normaalin Boundary2d:n työkaluilla ennen NEB-iteraation aloittamista.

3.2 Käyttöesimerkki

Jotta NEB:n tätä versiota voisi käyttää, pitää osata käyttää Boundarya. Tämä onnistuu kokeilemalla ja/tai lukemalla manuaalia. Tässä kappaleessa on tyypillinen NEB:n käyttöesimerkki, jonka lukemalla ja kokeilemalla NEB:n käytön pitäisi olla helppo aloittaa. Seuraavassa kappaleessa on esitetty kaikki NEB:n säätimet (engl. widget) yksitellen.

Osa NEB:n parametreista voidaan säätää vain Boundaryn parametritiedostossa "boundary.param". Näitä ovat mm. kuvien määrä sekä suppenemiskriteerit.

Alkukuva luodaan normaalisti käyttäen Boundaryn normaaleja säätimiä, esim.

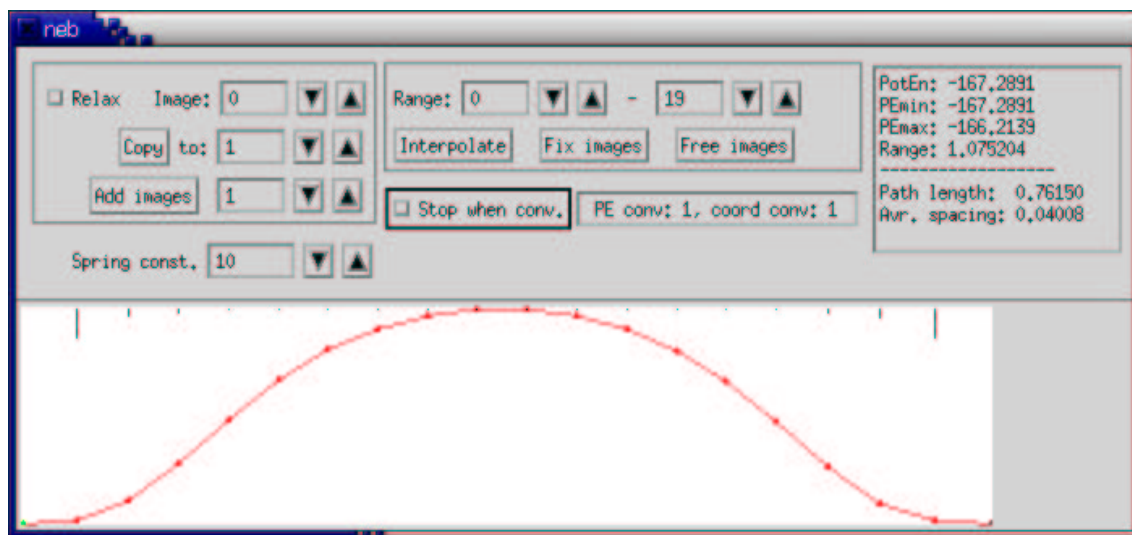
Temperature- ja **Modify Area-**dialogeja. Alkukuvan olisi hyvä olla relaxoitu, minkä voi suorittaa **Temperature-**dialogin **Quench-**napilla. Tämän jälkeen simulaatio pysäytetään (**Pause**) ja avataan NEB-dialogi. Tämän dialogin avaaminen kopioi ensimmäisen kuvan konfiguraation kaikkien muiden kuvien aloituskonfiguraatioiksi. Seuraavaksi vaihdetaan kuvanvalitsimella (**Image**) lopputilaan ja suoritetaan siihen tarvittavat muutokset, jotta se olisi halutun mukainen. Alku- ja loppukuvien identtisyys ei ole järkevää. Muutosten suorittamisessa tarvitaan todennäköisesti lopputilan relaxointia. Tämä tehdään painamalla **relax-**nappi alas ja sen jälkeen käynnistämällä normaali MD-simulaatio (**Run**), jolloin simulaatiota ajetaan normaalisti mutta ainoastaan valitussa kuvassa (eli siinä, jonka numero on **Image-**valitsimessa). Lopuksi interpoloidaan alku- ja lopputilojen välillä (**Interpolate**). Myös monimutkaisempia reaktiopolkua/aloituskonfiguraatioita voi luoda. Interpolointia voi suorittaa millä tahansa välillä ja mitä tahansa kuvaa voi erikseen muokata relaxoimalla tai modifioimalla jne.

Kun tilojen alustava konfiguraatio on valmis, otetaan relaksointi (**Relax**) pois käytöstä ja käynnistetään simulaatio pääikkunan **Run**-napilla. Tällöin ajetaan NEB-algoritmia. Kuvien määrä on kääntäen verrannollinen ajonopeuteen, joka on aina hitaampi kuin normaalissa Boundaryn MD-simulaatiossa.

NEB-dialogin kuvaaja-alueella näkyy potentiaalienergia polun funktiona. Kuvat on merkattu pienillä ympyröillä ja niiden välit vedetty yksinkertaisesti suoralla viivalla. Punainen ympyrä, tarkoittaa normaalia kuvaa, tummanpunainen kiinnitettyä kuvaa, sininen valittua eli näytössä olevaa ja vihreä valittua ja kiinnitettyä. Polun suoruus näytetään pystysuorilla mustilla viivoilla, jotka piirretään kuvaaja-alueen yläreunasta alkaen kuvien kohdalla. Jos viiva ylittää puoliväliin aluetta, on polussa suora kulma siinä kohdalla ja jos viiva ylittää aivan alareunaan asti, kääntyy polku 180 astetta, mikä on yleensä merkki huonosta toiminnasta tai liian pienestä kuvien määrästä. Täysin suora polku näkyy mustana pisteinä yläreunassa.

Toinen vaihtoehto kuvien alkutilan luomiseksi on tuottaa ne normaaleilla Boundary-ajoilla ja tallentaa ne tiedostoihin. Tämän jälkeen kuvia voidaan lukea tiedostosta NEB:hen yksi, muutama tai kaikki. Lukeminen onnistuu käynnistämällä Boundary -**readatoms** AAAA-parametrilla (AAAA on nelinumeroinen luku). Tallennettujen kuvien nimet ovat muotoa **coords.AAAA.iBBB.xyz**, missä AAAA on päänumero ja BBB kuvan numero. Ohjelma tallentaa aina kaikki kuvat valittaessa **Save Snapshot**, jos NEB on käytössä. Ohjelma myös lataa aina kaikki löytämänsä kuvat, jos NEB ja **-readatoms** ovat molemmat käytössä. Ladatut kuvat on kuitenkin helppo päällekirjoittaa **Copy**-painonapilla tarvittaessa. Kaikkia kuvatiedostoja ei ole pakko olla olemassa, missä tapauksessa niiden paikalle simulaatioon kopioidaan ensimmäinen kuva. Tämän jälkeen esim. interpolaatiolla voi muokata välikuvat mieleisekseen.

3.3 Säädinkohtaiset käyttöohjeet



Kuva 4: NEB-dialogi

- **Image** vaihtaa *valittua kuvaa*. Tähän kuvaan kohdistuvat tässä kehyksessä olevat napit ja tämä kuva piirretään pääikkunaan.
- **Relax** kytkee relaksointimoodin päälle/pois. Jos moodi on pois päältä, ajetaan NEB-algoritmia kaikille kuville. Jos moodi on päällä, ajetaan normaalia MD-simulaatiota

valitulle kuvalle, jolloin voidaan käyttää esim. **Temperature**-dialogin säätimiä. Tämä kontrolli on hyödyllinen alku- ja lopputiloja luotaessa.

- **Copy** kopioi *valitun kuvan to*-kuvan päälle.
- **Add images** lisää halutun määrän kuvia *valitun kuvan* ja seuraavan kuvan väliin.
- **Range** vaihtaa *valittua väliä*, johon väleihin kohdistuvat operaatiot (eli tämän kehyksen napit) vaikuttavat.
- **Interpolate** interpoloi *valitun välin* kuvien koordinaatit. Tämä on hyödyllinen esim. välikuvien luonnissa ennen ajoa, kunhan alku- ja loppukuvat on saatu tehtyä.
- **Fix images** kiinnittää *valitun välin* kuvat. Kiinnitetyt kuvat toimivat samoin kuin alku- ja loppukuva: niiden koordinaatit eivät muutu.
- **Free images** vapauttaa edellisellä napilla kiinnitetyt kuvat *valitulla välillä*. Tämän jälkeen ne toimivat taas aivan normaalisti eli koordinaatteja päivitetään NEB-algoritmin mukaisesti.
- **Stop when conv.** pysäyttää (mutta ei sulje ohjelmaa) NEB- algoritmin, kun kunkin kuvan koordinaatit ja potentiaalienergia muuttuvat yhdellä aika-askeleella vähemmän kuin **.param**-tiedostossa määritellyn suppenemiskriteerin verran.
- **PE conv.** -tekstikenttä näyttää, onko suppenemisehdot saavutettu. Jos ehto on saavutettu, luku on 1 ja jos ei ole saavutettu.
- **PE, PEmin** -tekstikenttä näyttää *valitun kuvan* potentiaalienergian, potentiaalienergian minimin ja maksimin sekä maksimin ja minimin erotuksen, polun kokonaispituuden ja kuvien keskimääräisen etäisyyden. Näistä tärkein tieto on usein potentiaalienergian maksimin ja minimin erotus, sillä monissa tapauksissa tämä määrää likimäärin reaktionopeuden.
- **Potentiaalienergian kuvaaja** näyttää potentiaalienergiakäyrän. Kuvat ovat pieniä ympyröitä ja niiden etäisyydet x-akselilla pohjautuvat niiden etäisyyteen toisistaan. Ideaalisessa tapauksessa kuvat ovat tasavälisesti, mutta käytännössä näin ei aina ole. Sininen ympyrä näyttää *valittua kuvaa*, vihreä ympyrä kiinnitettyä *valittua kuvaa*, punainen ympyrä normaalia kuvaa ja tummanpunainen ympyrä kiinnitettyä kuvaa. Mustat viivat yläreunasta alkaen kuvaavat polun suoruutta, oikeastaan kosinina polun kulmasta. Jos viiva on ihan yläreunassa, polku on ihan suora. Jos viiva on kuvaaja-alueen puolivälissä, polku tekee 90 asteen mutkan. Jos taas viiva ulottuu aivan kuvaaja-alueen alareunaan asti, polku tekee 180 asteen mutkan, mikä ei ole hyvä merkki. Polun kulman kosini määritellään täsmällisesti kaavalla

$$\cos(\alpha) = \frac{\tau_1 \cdot \tau_2}{|\tau_1||\tau_2|} \quad (8)$$

missä siis α on polun kulma tietyn kuvan kohdalla, τ_1 ja τ_2 suorat edelliseen ja seuraavaan kuvaan.

- **Spring const** määrää NEB:ssä käytettävän jousivakion. Tämän ei tarvitse olla vakio koko ajan aikana vaan sitä voi vapaasti muuttaa kesken ajonkin.

Näiden lisäksi NEB:llä on myös muutama muuttuja, joiden arvoa voi säätää parametri-tiedostoa editoimalla.

- `images` säätää kuvien alkuarvon. Kuvia voidaan myöhemmin lisätä.
- `nebk` säätää NEB:ssä tarvittavan jousivakion arvon.
- `nebcStop` säätää suppenemispysäyttimen alkuarvon (1 tai 0). Tätä voidaan simulaation aikana vaihtaa.
- `wnebPlotSize.x` säätää NEB:n kuvaaja-alueen leveyden.
- `wnebPlotSize.y` säätää NEB:n kuvaaja-alueen korkeuden.
- `nebec` säätää potentiaalienergian konvergoitumisehdon: kullekin kuvalle tulee päteä:

$$|(PE_{vanha} - PE_{uusi})/PE_{uusi}| < nebec$$
- `nebccc` säätää koordinaattien konvergoitumisehdon: kaikille koordinaateille tulee päteä ylläolevan kaltainen ehto.

4 Testiajojen tuloksia ja havaittuja ongelmia

4.1 Yleistä

Testiajoissa kokeiltiin erilaisia tapauksia, esim. miten jousivakio, kuvien määrä, alku- ja lopputilan etäisyys toisistaan, kiinnitetyt atomit ja erilaiset tapaukset vaikuttavat lopputulokseen.

Yksinkertaiset ja lyhyet testiajot, esim. irtoatomin siirtyminen pinnan päällä potentiaali-kuopasta toiseen, tuottivat oikean näköisiä tuloksia. Tulokset olivat hyviä myös, jos atomi siirtyi monta väliä.

Hankalampia kokeita olivat dislokaation siirtymiskokeet. Näissä esim. 40x40- kokoiseen kappaleeseen aiheutettiin atomirivin poistolla dislokaatio, jota termisesti tai ulkoisella voimalla siirrettiin yksi tai useampi atomirivi. Näissä kokeissa siirtyviä atomeita oli aina hyvin paljon. Tulokset olivat kohtuullisen hyvän näköisiä, jos dislokaatio siirtyi enintään muutama atomiriviä. Jos dislokaatio siirtyi vaikkapa 20 atomiriviä, keskeltä reunaan, tulokset olivat huonompia. Tämä johtunee siitä, että poluista tuli hyvin pitkiä ja monimutkaisia. Tällöin alun lineaarinen interpolaatio johti hyvin kauaksi minimienergiapolusta ja suppeneminen polulle oli hidasta tai mahdotonta. Jo neljän atomirivin kokoinen dislokaation siirtymän suppeneminen lineaarisen interpolaation luoman polusta minimienergiapolulle kesti erittäin kauan. Lisähankaluutena dislokaatiokokeissa oli se, että siirtymisen energiavalli oli varsin pieni. Hankalahkoilla erikoisjärjestelyillä saatiin parhaat tulokset näistä. Näissä järjestelyissä otettiin dislokaation siirtyessä normaalissa MD-ajossa (**relax**) kuvia talteen NEB:iä varten (**Copy**) varsin tiheään ja sitten ajettiin NEB-metodia sekä lisättiin kuvien määrä sopivaksi (esim. 200) ja oikaistiin polussa syntyneitä virheitä jne. Tästä ajosta tuli kuitenkin niin raskas, että sen ajaminen normaalilla tietokoneella voi kestää hyvin kauan, jopa viikkoja.

Erilaisilla jousivakioilla kokeiltaessa huomattiin, että pienet jousivakion arvot eivät riitä takaamaan tasavälistä jakaumaa ainakaan kohtuullisilla ajoajoilla ja sellaisissa tapauksissa, joissa alun interpolaatiopolku eroaa lopputuloksesta merkittävästi. Liian isot jousivakiot taas luonnollisesti johtivat systeemin hajoamiseen, jos aika-askelta ei lyhennetty, mikä taas olisi johtanut pitkään ajoaikaan. Monissa tapauksissa hyvä keino oli ajaa ensin pienellä jousivakiolla, jotta NEB suppenisi minimienergiapolulle ja sitten, välien tasaamiseksi kasvattaa jousivakiota suureksi.

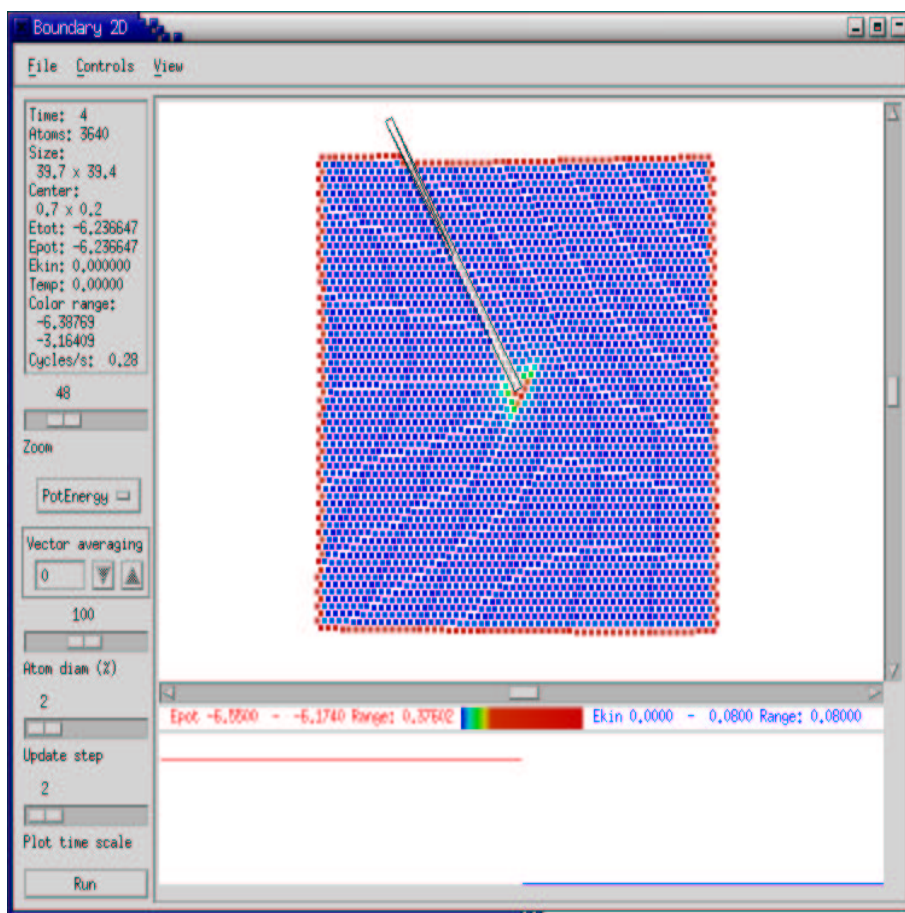
Sekä pienet että isot kuvamäärät näyttivät toimivan niin kuin pitääkin. Liian pienillä kuvamäärillä tilanteen suhteen oli tietysti se vaikutus, että polku jäi epätarkaksi ja pienet yksityiskohdat kadotettiin. Kuviahan voi tietysti aina lisätä sopiviin väleihin. Tässä tapauksessa kuvien tiheys voi muuttua epätasaiseksi jolloin vaihtoehtoina ovat: a) odottaa, että NEB tasaa tiheyden (voi kestää kauankin tilanteesta riippuen) b) kiinnittää joitakin kuvia, jolloin tiheys kullakin välillä jää vakioksi (voi johtaa huonoon lopputulokseen, jos kiinnitetyt kuvat eivät ole vielä supenneet) c) kasvattaa jousivakiota reilusti, jolloin NEB vetää polun tasatiheäksi melko nopeasti (toimii yleensä hyvin) d) lisätä kuvia siten, että kuvatiheys pysyy vakiona, joka on usein paras vaihtoehto, mutta voi johtaa kuvien määrän liiallisen kasvun johdosta hitauteen.

Alku- ja lopputilan etäisyyden (eli polun pituuden) vaikutuksesta voidaan todeta, että dislokaatioiden siirtymisessä pitkiä matkoja oli hankalia ongelmia (mutkainen polku, hidas suppeneminen jne), mutta muut kokeillut tapaukset (esim. atomin tai vakanssin siirtyminen) toimivat hyvin. Tämä johtunee siitä, että dislokaatioiden siirtymisessä melkein kaikki atomit liikkuvat ja energiavallit ovat matalia ja muissa kokeiluissa tapauksissa siirtyi vain vähän atomeja suhteellisen korkeiden energiavallien yli.

Erittäin tärkeä tekijä NEB:tä käytettäessä on kiinnitettävien atomien valinta. Tämä toteutetaan **Modify Area**-dialogin avulla. Kiinnityksien tarkoitus on estää NEB-algoritmia pyörittelemästä, siirtelemästä tai vääntelemästä koko systeemiä. Kiinnitykset pitää valita tilanteesta riippuen. Hyvän tuloksen saa, jos jossain päin systeemiä on alue, joka ei osallistu juurikaan reaktioon (esim. reuna), jolloin sen voi kiinnittää. Monissa tapauksissa, ei kuitenkaan kaikissa, pätee sääntönä ”mitä enemmän kiinnitettyä aluetta, sitä parempi”. Hankalammissa tapauksissa voi riittää kahden atomin kiinnitys (estää pyörimisen ja liikkumisen mutta ei muodonmuutoksia) tai ehkä jopa kiinnityksien poisjättäminen. Kokeilemalla samaan tilanteeseen erilaisia kiinnitysvaihtoehtoja paras vaihtoehto löytyy. Joissakin tilanteissa voi olla kätevää kiinnittää neljän atomin toinen koordinaatti, kaksi vaaka- ja kaksi pystykiinnitystä. Tämä kiinnittää saman määrän vapausasteita kuin kahden atomien täydellinen kiinnittäminen, mutta sallii systeemin relaxoitua enemmän.

4.2 Dislokaation liikkumisesimerkki

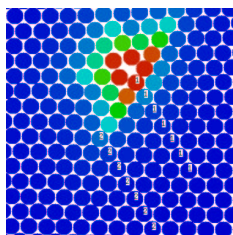
Tarkoituksena tässä esimerkissä oli tutkia dislokaation liikkumista kappaleen keskellä muutamien atomirivin verran. Tämä esimerkki myös havainnollistaa NEB:n käyttöä.



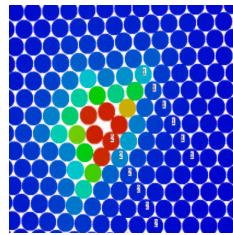
Kuva 5: Dislokaation synnyttäminen

Aluksi luotiin dislokaatio 40x40-kokoisen kappaleen keskelle. Dislokaatio luotiin poistamalla puoli atomiriviä kuvassa 5 näkyvästä kohdasta ja relaxoimalla (Quench to zero) syntynyt aukko. Seuraavaksi kaksi atomia kiinnitettiin (Modify Area -> Fix velocity: 0,0) vasemmasta ylänurkasta ja oikeasta alanurkasta. Tämän jälkeen tilanne oli kuvan 5 mukainen.

Seuraavaksi aukaistiin NEB-dialogi ja valittiin viimeinen kuva eli kuva 39 (joka siis oli kopio ensimmäisestä kuvasta). Valitsemalla **Relax** saatiin aikaan tilanne, jossa viimeistä kuvaa voitiin rauhassa muokata. Kuumentamalla kuvan kappale (**Temperature**) noin 30 Kelviniin dislokaatio lähti liikkeelle kohti vasenta alanurkkaa. Kun dislokaatio oli liikku-
nut noin neljän atomirivin verran, lopetettiin kuumennus ja aloitettiin relaxointi. Relax-
ointi johti dislokaation asettumiseen vakaasti neljännen atomirivin kohdalle ja lämpötilan
laskun. Kun järjestelmä oli relaxoitunut (eli lämpötila laskenut nolleen), suoritettiin in-
terpolaatio koko välillä 0-39. Kuvissa 6 ja 7 näkyy alku- ja lopputilanteiden oleellinen
alue eli kappaleen keskusta. Kuviin on merkattu ylimääräiset atomirivit alussa (merkki
1) ja lopussa (merkki 2). Kuvien värikoodaus on potentiaalienergian mukaan, punainen
merkitsee korkeampaa potentiaalienergiaa ja sininen matalampaa.



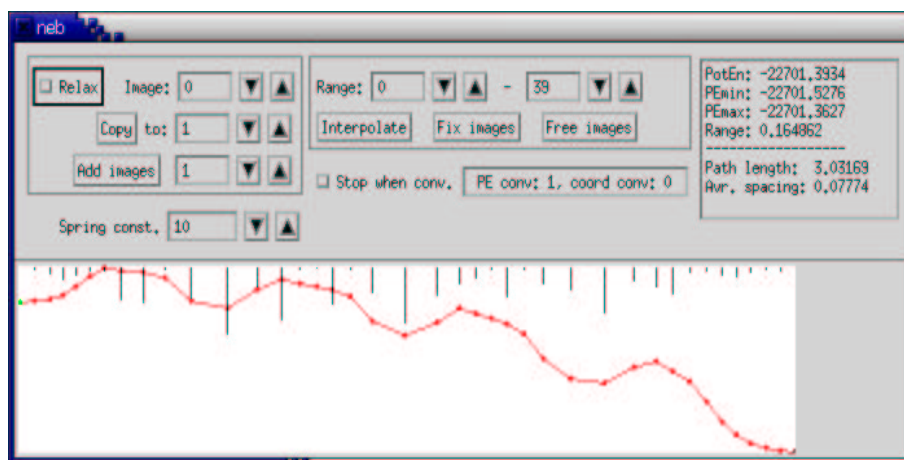
Kuva 6. Alkutilanne



Kuva 7. Lopputilanne

Seuraavaksi aloitettiin NEB-algoritmi kytkemällä **Relax** pois päältä. Pitkähkön ajon (useita tunteja) NEB oli saanut kuvat kohtalaisen lähelle minimienergiapolkua. Tulos näkyy kuvassa 8. Käyrässä vasen reuna on alkutilanne ja oikea reuna lopputilanne. Tästä huomataan, että NEB ei ole vielä täysin supennut (koordinaattien suppenemisindikaattori 0). Täydellinen suppeneminen on kuitenkin dislokaatioiden tapauksessa erittäin hidasta, joten tyydyttiin tähän.

Käyrää tarkastellessa huomataan, että dislokaation neljän atomirivin mittainen liike näkyy selvästi neljänä maksimina ja viitenä miniminä. Polku ei myöskään ole kovin mutkainen, mutta kuvat eivät ole ihan tasavälisiä. Tasavälisyyden olisi saanut aikaan pitemmällä ajoajalla tai suuremmalla jousivakiolla, mutta tämäkin tulos on riittävän tasavälinen. Lisäksi huomataan, että dislokaation liikkuminen on vahvasti sidoksissa reunaan: mitä lähempänä reunaa dislokaatio on, sitä pienempi on systeemin kokonaisenergia, joten dislokaatio liikkuu varsin helposti reunaa kohti pienellä lämmityksellä ainakin tämän kokoisessa (40x40) kappaleessa.



Kuva 8: Minimenergiapolku

5 Johtopäätöksiä

Tässä työssä oli päätarkoituksena testata NEB:tä mahdollisia tulevia realistisempia (3-D, paremmat potentiaalit jne) simulaatioita varten. NEB:iä koodatessa huomattiin, että se oli varsin helppo ja suoraviivainen implementoida **Boundary**-ohjelmaan tällä tavalla, joten se lienee helppo implementoida myös muihin ohjelmiin.

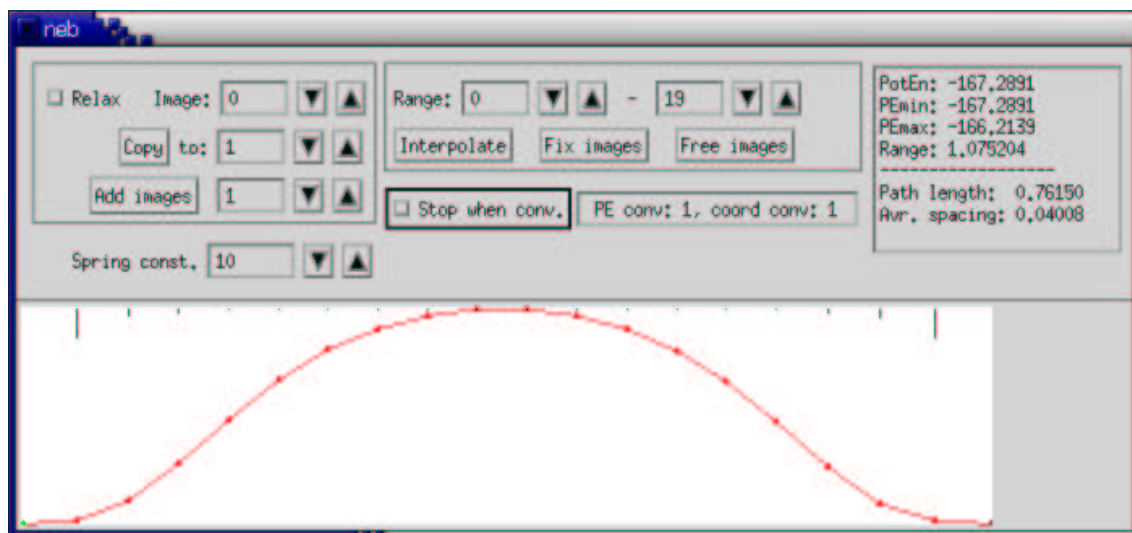
NEB selvisi kiitettävän hyvin sellaisista testeistä, joissa pääasiassa muutama atomia liikkuu ja muut pysyvät suurinpiirtein paikoillaan. Sen sijaan dislokaatiokokeista, joissa melkein kaikki atomit liikkuvat ja potentiaalivallit olivat matalia, NEB ei selvinnyt yhtä hyvin, mutta silti näistäkin saatiin joitakin hyviä tuloksia. Kuten edellisessä kappaleessa todettiin, liian kaukana MEP:stä oleva alkutilanne johti huonoihin tuloksiin. Ehkäpä NEB:n kykyä selvittää näistä hankalista tilanteista voisi tulevaisuudessa parantaa käyttämällä lineaarisen interpolaation sijaan jotain toista välikuvien muodostamismenetelmää. Tosin nykyinenkin versio sisältää joitakin mahdollisuuksia tähän, mutta ne ovat työläitä käyttää (esim. Copy).

Loppupäätelmänä voidaan sanoa, että NEB oli suhteellisen helppo implementoida, toimi hyvin useissa tapauksissa, mutta joutui ongelmiin liian hankalissa tilanteissa. Silti NEB:tä kannattanee testata jossain **Boundary**ä kehittyneemmässä MD-ohjelmassa.

A Using NEB

Generally, using the NEB method requires first setting the initial values for all images, then starting the actual NEB method and finally stopping it when it has converged. Sometimes it also requires fine-tuning during the run, for example adding images or fixing them.

The initial setup is easy to do: all images can either be read from files (name: `coords.XXXX.iYYY.xyz`, where XXXX is the general number which is same for all images in the same run and YYY is the image number which is different for all images in the same run) using `-readatoms` command line option or created interactively using normal Boundary controls. Especially useful in the 2nd approach are **relax**, **Interpolate**, **Quench to zero**, **Modify area** and **Copy** controls.



Kuva 9: The NEB dialog

- **Image** changes *the selected image*. Widgets in this frame affect this image and this image is also drawn in the main window.
- **Relax** toggles relaxing mode on/off. When this mode is on, (and the simulation is running), the normal MD simulation is run only in *the selected image*. This mode is useful for example in creating first and last images of the path prior to actual NEB run, e.g. using the **Temperature**-dialog: Quench etc. If this mode is off, NEB algorithm is run in all images.
- **Copy** copies *the selected image* to **to-image**, overwriting the previous coordinates of the **to-image**.
- **Add images** adds a chosen amount of new images between *the selected image* and the next one.
- **Range** changes *the selected range*, which is the range that is affected by range operations (i.e. widgets in this frame).
- **Interpolate** interpolates the coordinates of the images in *the selected range*. Very useful e.g. in creating the intermediate images prior to NEB run. Of course, the first and last images in the range must be created before this operation.
- **Fix images** fixes the images in *the selected range*. Fixed images work like the first and the last image in the NEB run: their coordinates won't change.
- **Free images** frees images fixed by the previous widget in *the selected range*. After this operation they are normal images whose coordinates are updated according to the NEB method.
- **Stop when conv.** pauses (doesn't quit Boundary) the program when coordinates and potential energies are converged according to the convergence parameters specified in **.param**.
- **PE conv. etc -text field** shows whether the convergence conditions have been met. 1 stands for met, 0 not met.
- **PE, PEmin jne. -text field** shows the potential energy of *the selected image*, minimum and maximum potential energies, the difference between max. and min potential energies (range), total length of NEB-path and the average distance between to images.
- **Potential energy plot** shows potential energy as a function of position in the path. Images are shown as small circles. In an ideal case the images are equally spaced but this is not always the case. A blue circle indicates *the selected image*, a green circle indicates a fixed and *selected image*, a red image indicates a normal image and a dark red image indicates a fixed normal image. The black lines starting from the upper border of the plotting area show the kinkiness of the path, actually a cosine of an angle on the path. If the line is short, just in the upper border, the path is straight. If the line reaches the middle point of the plotting area, the path has a right angle. If the line reaches the bottom of the plotting area, the path makes a 180 degrees turn which is not a good sign.
- **Spring const** changes the value of the spring constant used in NEB. This value doesn't have to be constant but it can be freely changed even during the NEB run.

In addition to these widgets, NEB also has a few variables which can be changed by editing the parameter (**.param**) file.

- `images` sets the initial number of images. Images can be added later but not removed.
- `nebk` sets the initial value of the sprint constant.
- `nebcStop` sets the initial value of "convergence stopper".
- `wnebPlotSize.x` sets the width of the NEB plotting area.
- `wnebPlotSize.y` sets the height of the NEB plotting area.
- `nebec` sets the potential energy convergence condition: NEB is converged according to potential energy when in each image $|PE_{previous} - PE_{current}| / PE_{current} < nebec$.
- `nebcc` sets the coordinate convergence condition: all coordinates must fulfill a condition similar to the above potential energy condition.

B Short Descriptions of NEB Functions and Procedures

This section contains very short descriptions of the main NEB functions.

- *MdCycle* (`simu.c`) is the controller function: it selects normal and NEB simulation modes and calls *nebCycle* if the NEB dialog is open.
- *nebCycle* (`simu.c`) calls *ComputeForces* for each image and then updates velocities and positions in each image by calling *nebUpdate* and *NebUpdatePositions* for each image. This function also handles single image relaxation.
- *nebdistance* (`simu.c`) calculates the distance between the given image and the previous image.
- *nebUpdate* (`simu.c`) is the most important NEB procedure. It calculates NEB spring forces and "nudges" actual forces and applies Verlet velocity projection in the current image (`cImage`).
- *NebUpdatePositions* (`simu.c`) moves atoms in the current (`cImage`) image.
- *OpenNebDialog* (`graphics.c`) opens the NEB dialog.
- *RedrawnebPlot* (`graphics.c`) redraws the neb plot but doesn't display the new plot.
- *RefreshnebPlot* (`graphics.c`) displays the updated plot (double buffering).
- *LoadImageFromFile* (`control.c`) reads the given image from a file overriding the image's previous coordinates.
- *nebCopy* (`control.c`) copies the first image to all other images - used in initialization without reading from files. *InterPolate* (`textttcallback.c`) interpolates image coordinates in the given range. *AddImage* (`textttcallback.c`) adds the given amount of images after the shown (`grImage`) image. *SaveMEP* (`textttcallback.c`) saves the minimum energy path to a file.

A more detailed documentation can be found in the actual code. Each function is briefly commented in the beginning of the function and also some major or complicated parts of the functions' actual bodies are commented.

Viitteet

- [1] Graeme Henkelman: Methods for Calculating Rates of Transitions with Application to Catalysis and Crystal Growth. University of Washington 2001.
- [2] Hannes Jónsson, Greg Mills and Karsten W. Jacobsen: Nudged Elastic Band Method for Finding Minimum Energy Paths of Transitions. Classical and Quantum Dynamics in Condensed Phase Simulations, ed. B. J. Berne, G. Ciccotti and D. F. Coker (World Scientific, 1998).
- [3] J.Merimaa, L.F.Perondi and K.Kaski,
An interactive simulation program for visualizing complex phenomena in solids,
Comp. Phys. Comm. **124** (2000) 60.
- [4] <http://www.lce.hut.fi/~akuronen/boundary2d/>