Simulaatioiden automatisointi The Pencil Code -projektissa

Eetu Pelkonen

2022 Laurea

Laurea-ammattikorkeakoulu

Simulaatioiden automatisointi The Pencil Code -projektissa

Eetu Pelkonen

Tietojenkäsittely

Opinnäytetyö

Kesäkuu, 2022

Laurea-ammattikorkeakoulu **Tiivistelmä**
Tietojenkäsittelyn koulutus
Tradenomi (AMK)

Eetu Pelkonen
Simulaatioiden automatisointi The Pencil Code -projektissa
Vuosi 2022 Sivumäärä 26

Opinnäytetyön tavoite oli selostaa Aalto-yliopiston tietotekniikan laitokselle ja The Pencil Code -projektille luodun simulaatioiden automatisointityökalun kehitysprosessi. Tarve työkalun kehitykselle syntyi parametrien manuaalisen asettamisen suuren aikavaateen vuoksi.

Opinnäytetyö kuvaa miten projekti määriteltiin, suunniteltiin, toteutettiin, testattiin ja dokumentoitiin. Opinnäyte työ kuvaa myös kirjoittajan oppimisprosessia uusien teknologioiden, kehitysympäristöjen ja -menetelmien parissa. Projektissa käytettyjä teknologioita olivat mm. Linux, Python ja Pythonin lukuisat kirjastot.

Lopputuote saatiin valmiiksi usean muodonmuutoksen jälkeen. Automatisointityökalu lukee käyttäjän antamasta syötetiedostosta parametrien arvot ja luo sen mukaan useita simulaatioita. Työkalu dokumentoitiin ja jaettiin The Pencil Code -projektin käyttöön onnistuneesti. Työkalun jatkokehitys alkaa kesällä 2022. Jatkokehityskohteita ovat datapostprosessointi ja diagnostiikat.

Asiasanat: automatisointi, data, python

Laurea University of Applied Sciences **Abstract**
Degree Programme in Business Information Technology
Bachelor's thesis

Eetu Pelkonen

Automation of simulations in The Pencil Code -project

Year 2022 Pages 26

Objective of this thesis was to document the development process of a simulation automation tool, which was created for the needs of Aalto University Department of Computer Science and The Pencil Code -project. Main need for development stemmed from the time budgeting necessities of having to manually set parameters for simulations.

This thesis describes how the project framework was set up, how the tool was designed, developed and documented. The thesis also describes the learning process of its writer in new technologies, development environments and -methods. Technologies used in the project were e.g. Linux, Python, and libraries of Python.

End product was finished after multiple iterations. The automation tool reads the parameter input from a user-given text file and then sets up the simulations accordingly. The tool was documented and shared with the beneficiaries successfully. Further development of the tool starts in the summer of 2022 and will focus on datapostprocessing and running of diagnostics.

Keywords: automation, data, python

Sisällys

1 Johdanto 6
2 Lähtökohdat 6
2.1 Toimeksiantajan esittely 6
2.2 Kehittämiskohteen kuvaus ja kehittämistavoitteet 7
2.3 Aihealueen rajaus 7
2.4 Keskeiset käsitteet 7
3 Tietopohja 8
3.1 Kehitysympäristöjen kuvaus 8
3.2 Käytetyt teknologiat 11
3.3 The Pencil Code 12
4 Kehittämismenetelmät 12
4.1 Viikoittaiset tapaamiset 12
4.2 Pariohjelmointi 13
5 Kehittämistyön toteutus 14
5.1 Tiedonhankinta ja uusien teknologioiden opettelu 15
5.2 Projektin määrittely 16
5.3 Suunnittelutyö 16
5.4 Sovelluksen toteutus 17
5.5 Sovelluksen testaus 20
5.6 Sovelluksen dokumentointi ja julkaisu 21
6 Yhteenveto ja jatkokehitys 21
7 Oman oppimisen arviointi 22
Lähteet 24
Kuviot 26

1 Johdanto

Toukokuussa 2021 aloitin työharjoitteluni Aalto-yliopiston tietotekniikan laitoksella. Työharjoitteluni tavoite oli luoda automatisaatiotyökalu tieteellisissä tutkimuksissa käytettyjen simulaatioiden rakentamiseen. Tämä työ käsittelee mistä tarve sille tuli, miten sovellus suunniteltiin, mistä teknologioista sovellus koostuu, ja miten sovellus toteutettiin.

2 Lähtökohdat

Lähtökohdat projektityölle tulivat tarpeesta luoda työkalu, jolla helpottaa simulaatioiden ajamista lyhentämällä niiden järjestämiseen käytettyä aikaa. Tarjous toimeksiantajalta tuli huhtikuussa 2021 ja työskentely projektin parissa käynnistyi toukokuussa. Päätökseen se saatettiin saman vuoden marraskuussa. Projektityö tehtiin osana työharjoittelua Aalto-yliopiston Tietotekniikan laitoksella.

Lähtötasoni Python-ohjelmoinnin saralla oli alhainen. Olin ohjelmoinut ennen projektityötä ohjelmointikielistä lähinnä vain Javalla, joten uuden kielen opettelu projektikehityksen rinnalla toi lisähaasteen. Projekti tehtiin Linux-ympäristössä; minulle myös uusi alue.

2.1 Toimeksiantajan esittely

Toimeksiantajat projektille olivat Aalto-yliopiston tietotekniikan laitoksen astroinformatiikan ryhmä ja The Pencil Code -projekti. Espoossa sijaitseva Aalto-yliopiston tietotekniikan laitos on yksi johtavista Pohjois-Euroopan tietotekniikan tutkimuksen yksiköistä ja on yliopiston suurin laitos. (Aalto-yliopisto 2022.)

Tietotekniikan laitoksen alla toimiva Astrofysiikan ryhmä kehittää korkeatasoisia laskentatyökaluja kompleksien astrofyysisten järjestelmien simuloimiseen ja analysoimiseen. Heidän tutkimusaiheisiinsa kuuluvat mm. pyörteiset virtaukset, Aurinko, sen korona ja tähtienvälinen aine galakseissa. Heidän tämän hetken päätutkimustavoitteensa on ymmärtää paremmin Auringon magneettikentän vaihtelua ja pyrkiä estämään siitä koituvia vahinkoelementtejä. Heidän kehittämänsä laskentatyökalut käyttävät simulaatioissaan grafiikkaprosessorijärjestelmiä ja data-analysoinnissaan koneoppimista. (Aalto-yliopisto 2022.)

2.2 Kehittämiskohteen kuvaus ja kehittämistavoitteet

Suuri hidaste The Pencil Code -projektin kanssa työskenteleville henkilöille oli parametrien manuaalinen asettaminen ennen jokaista simulaation ajoa. Se vei turhaa aikaa tutkimustyöltä, joten ratkaisu asetustyön automatisoimiseksi tarvittiin. Sekundääritavoite projektille oli luoda diagnostiikoita ja muuta postprosessointia jokaisesta ajetusta simulaatiosta.

Valmis automaatiotyökalu tulisi keskeiseksi osaksi Pencil Code -projektia ja jokaisen käytettäväksi.

2.3 Aihealueen rajaus

Opinnäytetyö keskittyy kertomaan minkälainen kehitetty automatisointityökalu on, mitä teknologioita opin käyttämään ja mitä teknologioita kehitystyössä käytettiin. Se kertoo myös mikä The Pencil Code on, miksi työkalua tarvittiin, miten projekti kehitettiin, ketkä sitä kehittivät, mikä tekijän tietopohja on, miten se toteutettiin ja mihin se julkaistiin.

Opinnäytetyö ei kerro yksityiskohtaisesti siitä, mitä lopullisesta projektista jätettiin pois, sen keskeneräisistä kokeiluversioista tai miten debugaus tehtiin.

2.4 Keskeiset käsitteet

Python — Avoimen lähdekoodin korkeatasoinen, mutta samalla yksinkertainen ja ohjelmointikieli, joka on suunniteltu mahdollisimman helpoksi lukea ja ymmärtää.

Linux — Avoimen lähdekoodin käyttöjärjestelmä, jonka kehittämiseen jokainen voi osallistua, erittäin modulaarinen ja skaalautuva, supertietokoneissa käytetty.

The Pencil Code

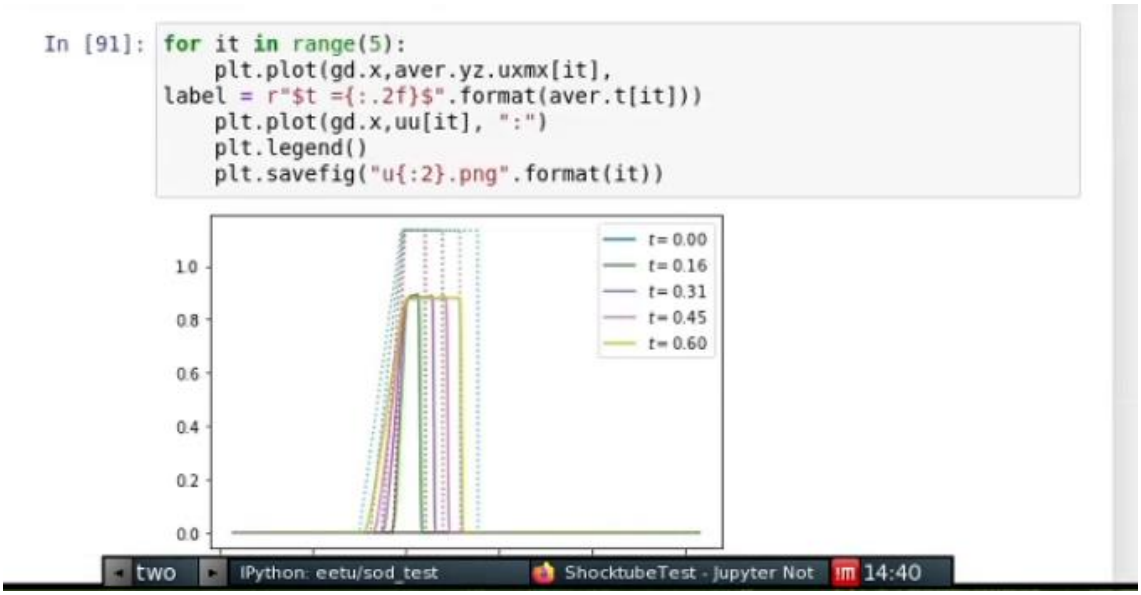
Fortran 95:llä kirjoitettu matemaattisia menetelmiä käyttävä ohjelma, jolla kuvataan kompressoituja hydrodynaamisia virtauksia magneettisilla kentillä simulaatioita käyttämällä.

3 Tietopohja

Tietopohjana toimi entuudestaan melko rajoittunut tieto ohjelmoinnistani, joka jäi web-kieliin ja Javaan ennen projektin alkua. Onnekseni Python on kuitenkin aloittelijaystävällinen kieli, jonka omaksuin melko nopeasti. Tässä auttoi pariohjelmointi ja tutorointi esimieheni Frederick Gentin kanssa.

Asetimme aluksi virstanpaaluja kehitykselleni Python-ohjelmoijana; pieniä ohjelmointitehtäviä ja tutustumista The Pencil Coden simulaatioiden ajoon manuaalisesti. Varsinkin alussa testasimme, miten luoda visualisointeja simulaatioiden tuloksista.

Yksi näistä oli Sod shock tube -ongelma (Kuvio 1), yleinen vertailuanalyysitesti laskennallisille ohjelmille, jotka yrittävät simuloida mm. räjähdyksiä. (The Visual Room, 2022.) Alla olevassa kuviossa loin Jupyter Notebookissa aikaleimakuvaajaohjelman *t* määrittääkseni simulaation painetta eri ajankohdissa, joka sitten tallensi sen PNG-kuvatiedostoksi ja nimesi sen ohjelmassa annetun formaatin mukaisesti.



Kuvio 1: Yksinkertainen harjoitus graafisen esityksen luomisesta

3.1 Kehitysympäristöjen kuvaus

Linux on ilmainen avoimen lähdekoodin käyttöjärjestelmä, jonka kehittämiseen jokainen voi osallistua. (Red Hat, 2018.) Linuxia käyttää suurin osa maailman supertietokoneista, mukaan luettuna projektissamme käytetyt supertietokoneet, Puhti sekä Mahti, joita ylläpitää suomalainen Tieteen tietotekniikan keskus. Tieteen tietotekniikan keskus on Suomen valtion omistama ICT-ratkaisuja tarjoava instituutio. (CSC – Tieteen tietotekniikan keskus, 2022.)

Linuxin modulaarinen rakenne antaa hyvän pohjan supertietokoneille, jotka pystyvät muokkaamaan sitä lähes loputtomasti tarkoituksiinsa, muun muassa suorituskyvyn maksimoimiseen. Linux-kernelin koko skaalautuu myös hyvin – kerneliä voidaan ajaa kooissa 2Mb:stä 1 teraan. Avoimen lähdekoodin projektina Linux saa myös tietoturvapäivityksiä heti vian tapahtuessa, jotta voidaan varmistaa toiminnan turvallinen jatkuminen. (Zinoune, 2022.)

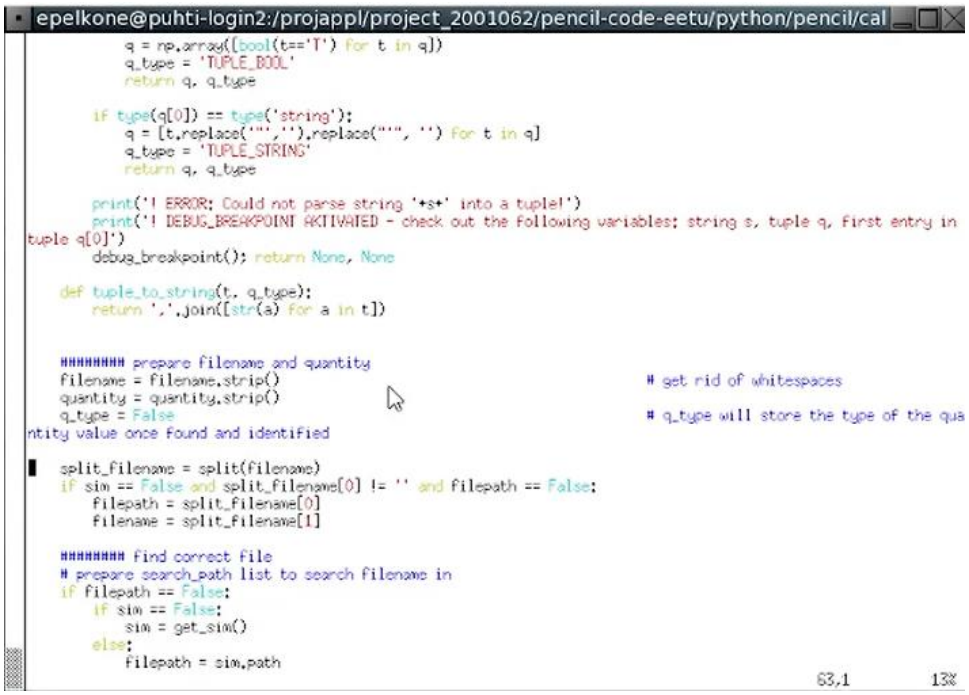
Projektikehityksessämme käytimme Nomachine-ohjelmaa etäyhteyden luomiseen supertietokoneissa sijaitseviin Linux-ympäristöihin. Linux-ympäristöä ei siis asennettu missään vaiheessa lokaalisti omalle tietokoneelle.

Vi (Kuvio 2) ja Emacs olivat kaksi mahdollista Linux-ympäristössä sijaitsevaa tekstieditoria. Emacsin valttina on sen graafinen käyttöliittymä, joka toimi kenties käytännöllisempänä ja helppokäyttöisempänä Windows-käyttöjärjestelmiin tottuneelle käyttäjällä. Esimieheni kuitenkin suositteli minulle käytettäväksi Vi:tä, perustellen sitä sillä, että ohjelmointi tapahtuu sulavammin ja nopeammin käyttämällä sen pikakomentoja, kunhan niiden käytössä harjaantuu, mistä olin lopulta samaa mieltä.

Vi’tä käyttäessä ohjelmoijan ei tule koskeakaan hiireensä, joka tuntui itsestäni aluksi hieman epäintuitiiviselta, mutta huomatessani nuolinäppäimien ja home/end-näppäinten, sekä muiden Vi’n omien komentojen sulavuuden, olin myyty. Esimerkkejä Vi-pikakomennoista, jotka tekevät ohjelmoinnin erittäin nopeaksi ovat:

”:rivinumero”, eli rivinvaihto, jolla navigoit koodia silmänräpäyksissä

”yy”, jolla kopioit rivin, missä olet. Käytetty yhteydessä komennon ”p” kanssa, joka sitten liittää sen haluamaasi paikkaan.

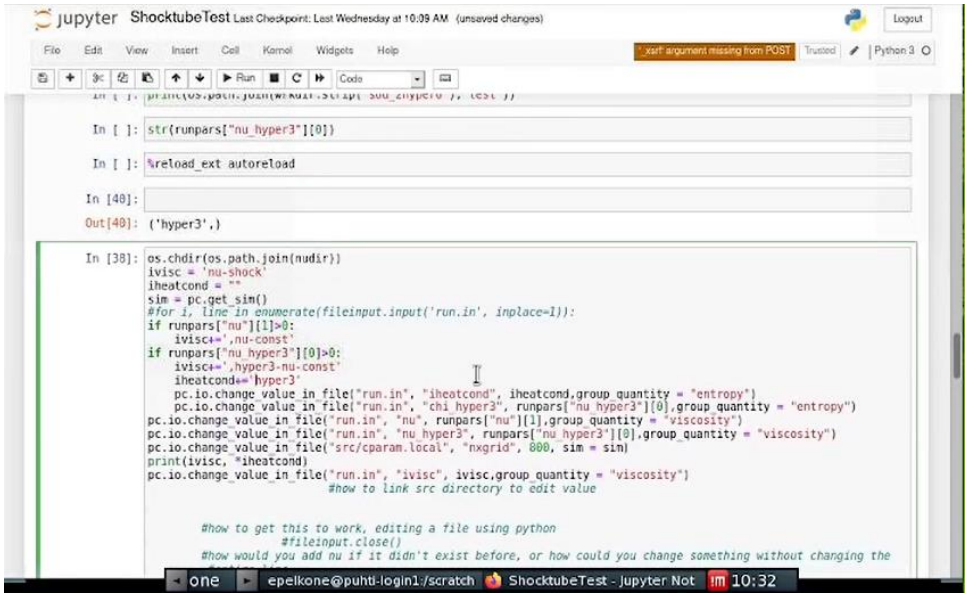


Kuvio 2: Vi-kehitysympäristö

Karhunosa ohjelmointityöstä suoritettiin kuitenkin Jupyter Notebook –moduulissa, joka muiden työssä käytettyjen moduulien tavoin asennettiin erikseen Linux-ympäristöön käyttämällä komentoriviä.

Jupyter Notebook (Kuvio 3) on selainpohjainen kehitysympäristö, joka muistuttaa tyypillisiä Windows-sovelluskehittimiä. Se on Interactive Python (Ipython) -moduulin haaraprojekti, jonka tarkoitus oli luoda käyttäjäystävällisempi ohjelmointiympäristö tyypillisestä shell-ikkunasta poiketen, ja jota voitaisiin käyttää muidenkin kielten kuin Pythonin kanssa ohjelmoinnissa. (Jupyter Notebook, 2015.)

Kehitystyössä ja Python-ohjelmoinnin opettelussa, Jupyter Notebookin antamat virhekoodit auttoivat löytämään viat koodissa, toiminto, jota ei esimerkiksi löytynyt arkaaisen minimalistisesta Vi’stä. Ohjelmointiprosessin aikana testasin ja loin Jupyterissä suurimman osan koodista, jotka lopulta siirsin valmiiksi Python-tiedostoiksi Vi:n kautta.



Kuvio 3: Jupyter Notebook -kehitysympäristö

3.2 Käytetyt teknologiat

Python on Python Software Foundation -säätiön hallinnoima avoimen lähdekoodin korkeatasoinen ja kompleksi, mutta samalla yksinkertainen ohjelmointikieli, joka on suunniteltu mahdollisimman helpoksi lukea ja ymmärtää. Ennen kehitysprojektin alkua en ollut käyttänyt Pythonia, mutta tästä syystä pääsin tyydyttävälle tasolle siinä melko nopeasti. Sen suosio perustuukin pitkälti sen lähestyttävyyteen ja Python-kehitysyhteisön aktiivisuuteen. Pythonilla on vahva asema erilaisissa käyttöympäristöissä, kuten frontend-kehityksessä, ylläpidossa ja testauksessa. (Python Suomi ry, 2022.)

Projektissamme käytetty ohjelmointikieli oli Python. Kaikki kehitystyö mitä luotiin tapahtui Pythonilla, vaikkakin esimerkiksi Fortran oli myös yksi The Pencil Code –projektissa käytetyistä kielistä.

Pythonin etu muihin ohjelmointikieliin tieteellisessä laskennassa ovat sen laajat kirjastot, mutta keskeisin syy on se, että sitä voidaan ajaa komentoriviltä suoraan, mikä on tärkeää, sillä työskentelimme Linuxin parissa. (MOOC, 2021.)

Python-kirjastoista käytimme projektikehityksessämme mm. NumPyä (Numerical Python), joka projektina tähtää tehostamaan numeraalista laskentaa Python-kielellä (NumPy.org, 2022.), ja pandasia, joka on työkalu datan analysointiin ja muokkaamiseen. (pandas, 2022.)

3.3 The Pencil Code

The Pencil Code on Fortran 95:llä kirjoitettu differenssimenetelmiä käyttävä ohjelma, jolla kuvataan kompressoituja hydrodynaamisia virtauksia magneettisilla kentillä simulaatioita käyttämällä. Ohjelma on hyvin modulaarinen ja sitä voidaan tarpeen vaatiessa soveltaa monissa erilaisissa konteksteissa. (The Pencil Code, 2022.) Se on kuitenkin eniten käytetty astrofysiikan tieteen laskennoissa, jonka alueella Aallon työryhmäkin toimi.

Yleisimmin ohjelmaa käytetään sen raskaiden laskentojen vaativuuden vuoksi massiivisissa rinnakkaisuutta hyödyntävissä supertietokoneissa, mutta se voidaan tehdä toimivaksi myös yhdellä tavallisella kotitietokoneella. Ensimmäisen kerran, ja milloin sen kehityksen voidaan sanoa alkaneen, on vuonna 2001, kun sitä edelleen kehittävät Axel Brandenburg ja Wolfgang Dobler käyttivät sitä simuloidakseen magnetohydrodynaamista propulsiota. (Brandenburg & Dobler, 2002.)

4 Kehittämismenetelmät

Projektityöskentelyssä työskentelin suurimmaksi osaksi yksin, noin 90% ajasta. Työjaksoni aluksi tapasimme esimieheni kanssa useammin, kun olin vielä opetteluvaiheessa teknologioiden kuten Linuxin ja Pythonin saralla.

Tapaamisemme silloin olivat suurimmaksi osaksi pariohjelmointia, missä jaoin näyttöäni Zoom-ohjelman kautta, ja katsoimme yhdessä erilaisia ongelmia ja kävimme läpi tehtäviä, joita hän jätti minulle ratkaistavaksi jokaisen tapaamisen jälkeen kehittääkseni ohjelmointitaitojani.

Tapasimme kuitenkin muuten kahdessa viikoittaisessa palaverissa, joista toinen oli ns. Python-ryhmä, jossa seuraamme liittyi kaksi muuta The Pencil Coden Python-kehittäjää. Toinen taas oli astrofysiikan aiheisiin kehittyvä viikoittainen Pencil Code -palaveri, jossa käytiin enemmän läpi tutkimuksia, joita The Pencil Coden kautta luotiin. Näissä palavereissa kävimme ringissä jokainen läpi vuorotellen, minkä parissa olimme työskennelleet kuluneen viikon aikana. Näissä tapaamisissa esiteltiin myös usein isompia, tunteja kestäviä, esityksiä, jotka saattoivat liittyä mm. tohtoriopintoihin.

4.1 Viikoittaiset tapaamiset

Viikoittaiset tapaamisemme The Pencil Code –projektiin jaettiin Python-ryhmään ja yleisempään tieteellisemmän työn työryhmätapaamiseen. Edellämainittuun ryhmään kuuluivat minä

ja esimieheni Frederick Gent sekä kaksi muuta Aalto-yliopistolla työskentelevää sekä opiskelevaa henkilöä. Jälkimmäisempään ryhmään taas osallistujia oli vaihdellen noin 10:stä 30:een, riippuen käsiteltävistä asioista.

Python-ryhmässä työskenteleminen olikin enemmän ”käytännön tekemistä”, isompaan ryhmään verrattuna. Vertailimme toistemme töitä, mutta käytimme suurimman osan ajasta minun ja esimieheni projektin läpikäymiseen ja kehittämiseen. Isoksi avuksi osoittautui, ja jota saamme kiittää monista kehitysideoista, Aallon tietotekniikan laitoksella asiantuntijan nimikkeellä työskentelevä Simo Tuomisto. Pythonin parissa paljon työskennelleenä hän osasi nimetä monia kirjastoja, jotka hän näki hyödyllisinä käytettäviksi projektissamme, joita sitten kokeilimme ja otimmekin käyttöön lopputuotteeseemme. Näitä olivat mm. NumPy ja Pandas.

Viikoittaisessa yleisemmässä The Pencil Code –tapaamisessa keskustelu oli pääosassa. Kävimme Zoom-kierroksella kaikkien viikoittaisia aikaansaannoksia läpi, jotka yleensä koskivatkin tieteellisiä julkaisuja, joiden parissa tohtoriopiskelijat ja postdoc-tutkijat työskentelivät. Toisin kuin Python-ryhmämme, TPC-tapaaminen oli erittäin kansainvälinen ja käsitti tutkijoita ja opiskelijoita eri yliopistoista ympäri maailmaa. Suurin osa heistä tuli eurooppalaisista tutkimusinstituuteista, kuten Tukholman NORDITAsta ja Newcastlen yliopistosta, josta oma esimiehenikin Aallolle oli siirtynyt. Kuten aiemmin mainittu, tieteellisten julkaisujen aiheet käsittelivät lähes poikkeuksetta astroinformatiikan ja –fysiikan aiheita, mutta myös Pencil Coden kehitystä. Omalla vuorollani esittelin lyhyesti, miten projektini oli edistynyt ruudunjakodemonstraatiolla. Vaikka ajoittain - esimerkiksi työharjoitteluni alkupuolella - ne olivat melko yksinkertaisia, he olivat ymmärtäväisiä ja loivat luottamusta työhöni - en koskaan kokenut paineita esittämisestä.

Jätin osallistumisen tapaamisiin kuitenkin harvemmalle jonkin ajan jälkeen työharjoittelussani, koska ne usein kävivät läpi asioita, jotka eivät minulle olleet relevantteja, ja keskityin sen sijaan omaan työhöni.

4.2 Pariohjelmointi

Pariohjelmointi on työskentelymenetelmä, jossa nimensä mukaisesti ohjelmoijapari tekee yhteistyötä, missä toinen tuottaa koodia, ja toinen seuraa vierestä, antaen neuvoja ja ehdotuksia ohjelmointiin liittyen. Rooleja tulisi vaihdella säännöllisesti aika ajoin.

Yksi esimerkki pariohjelmointimenetelmistä on Extreme Programming (XP), jossa toimitaan lyhyiden kehityssyklien sisällä, ja julkaistaan usein uusia iteraatioita, joita voidaan käsitellä tavoitteiden raameissa. Iteraatioiden pituus on noin kaksi viikkoa. Pareina näin toimiessa voi-

daan antaa palautetta toiselle helposti, kehittää ymmärrystä koodista helpommin ja refaktoroida sitä helposti. (Turun yliopisto, 2015.)

Pariohjelmointi säilyi työssämme yleisenä läpi koko projektikehityksen aikana. Kuten mainittu, aluksi tapasimme useammin, koska olin vielä noviisi työssä käsiteltävissä aiheissa ja käytetyissä teknologioissa. Alun tapaamisissamme minulle saneltiinkin usein sanalleen mitä ohjelmoida ja kirjoittaa, emmekä niinkään vaihtaneet rooleja.

Koronapandemian vuoksi pariohjelmointi toteutettiin täysin etänä, kuten koko työharjoittelujaksoni, vaikka työskentelin pääosin Aalto-yliopiston kampuksella työskentelevien henkilöiden kanssa, The Pencil Code -projektin kansainvälisyydestä huolimatta.

Jonkin ajan jälkeen olin kuitenkin niin tottunut kehitysympäristöihimme ja käytettyyn kieleemme, että pystyin työskentelemään yhä itsenäisemmin, jolloin tapaamisemme harventuivat projektikaaren keskivaiheilla, ja silloinkin tavatessa pystyin kontribuoimaan oman osani. Projektin lopussa tapaamisemme kuitenkin taas yleistyivät sillä sen viimeistelyssä, dokumentoinnissa ja julkaisemisessa oli minulle jälleen uutta opittavaa.

5 Kehittämistyön toteutus

Kehittämistyön toteutus alkoi lähes heti työharjoitteluni alkaessa vuoden 2021 toukokuussa. Siirtymä ohjelmointiharjoitusten ja varsinaisen lopputyön kanssa työskentelemiseen oli melko harmaa, koska harjoitukset koostuivatkin osaksi lopputyöhön sisälletyistä osista.

Ihan aluksi harjoitukset koostuivat Sod shock tube –ohjelman yhdistämisestä Jupyter Notebook -työskentely-ympäristööni ja kuvaajien luomisesta simulaatioista. Sitten opettelin vaihtamaan manuaalisesti simulaatioiden parametrejä, esim. kyseisessä ohjelmassa, joka oli sitä ennen kopioitu omaan hakemistooni supertietokone Puhdin sisällä, joten pystyin muuttaa versiotani ohjelmasta haluni mukaan (Kuvio 4).

Kuvio 4: Sod shocktube –ohjelman parametrinäkymä

Projektin keskivaiheilla yleisten kesälomaviikkojen aikana kesä-heinäkuussa työskentelin yksin pelkästään projektin parissa. Tällöin muutkin viikoittaiset palaverit loppuivat lähes kokonaan.

Postprosessoinnin eli esimerkiksi kuvaajien luomisen kehittämisen lopetimme noin heinäkuun lopulla riittämättömien tulosten ja kehityssuunnan puutteen vuoksi, ja käänsimme keskittymisen itse automaatio-ohjelman luomiseen. Tällöin koin itse ohjelmoinnin vaikeustason nousseen merkittävästi, sillä pelkkä kuvaajien luominen oli melko yksinkertaista.

Aloin tällöin kehittämään erilaisia kokeiluja automaatioratkaisun luomiseen. Niitä oli kehitystyön aikana useita, yhden suuren ohjelman luomisesta, moneen pienestä aliohjelmasta koostuvaan.

Elokuun aikana yhteinen työskentelytahtimme myös kiristyi. Tapasimme projektikatsauksen parissa nyt viikon aikana kahdesti yhden sijaan. Aloimme tässä vaiheessa pilkkoa työympäristöstäni kehittämiäni ratkaisuja omiksi ohjelmikseen, jotka sitten lopulta muodostivat loppukokonaisuuden ja toimivan ohjelman. Projektimme oli valmis 1. marraskuuta.

5.1 Tiedonhankinta ja uusien teknologioiden opettelu

Uuden tiedon omaksuminen oli yleistä harjoitteluni aikana. Kuten aiemmin mainittu, Python ja useat muut teknologiat olivat minulle entuudestaan melko tuntemattomia, vaikka minulla olikin taustaa ohjelmoinnin parissa.

Tutustuin aluksi The Pencil Code –projektiin. Sen täydellinen manuaali (The Pencil Code, 2022.) ja Quick Start –ohje (The Pencil Code, 2021.) olivat käytössäni useaan otteeseen projektin aikana. Quick Start –ohjeesta opettelin miten asentaa The Pencil Code työympäristööni, ja miten konfiguroida shell-ympäristö sille oikeaksi. Ohjeen kautta opin myös, miten luoda ensimmäinen simulaatioajoni. Täydellinen Pencil Code –manuaali olikin 300-sivuinen opus, joka käsitti kaiken simulaatioiden yksityiskohtaisesta kuvaamisesta ja laskentokaavoista parametrien tarkoituksiin.

Itse ohjelmointityössäni suurimmaksi tiedonhankintakanavaksi osoittautuivat pariohjelmointi- ja Python-ryhmä-tapaamisemme. Koen oppivani parhaiten tekemällä ja keskustelemalla aiheista opinkin eniten uutta. Huomasin, että varsinkin Python-ryhmän tapaamisten jälkeen sain edistyttyä työssäni eniten, koska muiden uuden näkökulmat antoivat minulle uusia perspektiivejä ongelmiin.

Toinen ohjelmointityössä usein käytetty tiedonhankintakanava oli hakukone Google. Ongelman kohdatessa sen sijaan, että olisin pohtinut itse ratkaisua, käännyin heti hakukoneen apuun. Loogistahan on, että etsisi valmista oikeaa vastausta ongelmaan, jos se on olemassa, eikä käyttäisi aikaa turhaansa sen tuijottamiseen. Yleisimpiä sivustoja, joista löysin apua, olivat esimerkiksi Python-moduulien omat dokumentaatiot, tai StackOverflow- ja Reddit-keskustelupalstat.

5.2 Projektin määrittely

Automaatio-ohjelman käyttötarkoitusperien taustalla minulla ei ollut yhtään informaatiota. Tiesin vain sen, että simulaatioiden ajo vei liikaa aikaa tutkimustyöltä työntekijöiden parissa, ja ratkaisu tälle tarvittiin.

Suunnittelutyössä siis luotin täysin esimieheni antamaan informaatioon siihen, mitä tultiin tarvitsemaan. Hän määrätietoisuutensa lisäksi kuitenkin oli yksi näitä simulaatioajoja ahkerasti käyttävistä henkilöistä, joten olin koko kehitysprojektin ajan varma, että liikuimme oikeaan suuntaan. Tietenkin projektia esiteltiin Pencil Coden viikoittaisessa tapaamisessa, joten kaikki tiesivät missä vaiheessa projekti oli, ja mitä se sisälsi.

5.3 Suunnittelutyö

Alkuperäisessä projektisuunnitelmassa tavoitelinjaukset olivat luoda työkaluja, jotka helpottaisivat useaparametristen simulaatioiden valmistelua ja käynnistämistä. Työkalujen tarkoitus

oli olla suunniteltu niin, että niille pystyi antamaan lukuisia ehtolauseita ja argumentteja, ja toimia minkä tahansa Pencil Coden simulaatio-objektin kanssa.

Alkuperäisessä projektisuunnitelmassa oli tarkoitus myös luoda työkaluja diagnostiikkojen ja datapostprosessoinnin visualisoimiselle ja luoda MPI-versio valmiista työkalusta.

MPI on ohjelmointirajapinta, jonka kautta Python-ohjelmat voivat käyttää hyväksi useita prosessoreita esimerkiksi työasemissa, klustereissa ja supertietokoneissa. Se on laajimmin käytetty laskennassa, jossa useat prosessointiyksiköt toimivat rinnakkain. (Open MPI, 2022.)

5.4 Sovelluksen toteutus

Sovellus koostuu useasta aliohjelmasta. Sovelluskehityksen alussa yritimme luoda yhtä suurta ohjelmaa, mutta päätimme lopulta jakaa projektin neljään pienempään aliohjelmaan. Sovelluksen ja sen polun nimeksi Pencil Coden sisällä tuli “Pipelines”, ja se koostuu ohjelmista: “Clonesimulations”, “parameter_table”, “trim_table” ja “make_sims_dict”. Lyhyempi ohjelma, jota kutsutaan “initiks”, lukee datan ja parametrit sille annetuista datahakemistoista.

Clonesimulations (Kuvio 5) luo ryhmän hakemistoja sille annetusta simulaatio-objektista, joka sisältää listan parametreistä, joita tulee monistaa simulaatioita varten. “Simset”-muuttuja on käyttäjän antama parametrihakemisto ja “simsdir” polku, johon valmiit simulaatiot sitten luodaan. Ohjelman tuloste on ryhmä kääntämättömiä ja ajamattomia simulaatioajohakemistoja, joiden parametrit ovat muutettu käyttäjän haluamiksi.

```
Examples
--------
>>> simsdir = '/path/to/set_of_clones'
>>> params = pencil.pipelines.parameter_table('example_filename.txt')
>>> params = user_own_trim_table(params)#See pc.pipelines.trim_table
>>> simset = pencil.pipelines.make_sims_dict(params)
>>> clone_sims(simset,simsdir=simsdir)
"""
#If user provides no clone path
if not simsdir:
    simsdir = os.getcwd().strip(os.getcwd().split('/')[-1])
mkdir(simsdir)
#For each set of simulation parameters create new simulation subdirectory
for sim in simset:
    newdir = join(simsdir,sim)
    cmd = ['pc_newrun',"-s",newdir]
    #Only compile if makefile.local or cparam.local change
    if "compile" in simset[sim].keys():
        if simset[sim]['compile']:
            cmd = ['pc_newrun',newdir]
    process = subprocess.Popen(cmd, stdout=subprocess.PIPE)
    process.communicate()
    for filename in simset[sim]:
        #'compile' flag only used above
        if not filename == 'compile':
            #Files which are f90nml-compatible
            if "run.in" in filename or "start.in" in filename:
                pars = f90nml.read(filename)
                newpars = pars.copy()
                for group in simset[sim][filename]:
                    for item in simset[sim][filename][group]:
                        newpars[group][item] =  simset[sim][filename][group][item]
                newpars.write(join(newdir,filename),force=True)
            else:
                file_path = newdir
                if "local" in filename:
                    file_path = join(file_path,'src')
                for item in simset[sim][filename]:
                    change_value_in_file(filename, item,
                            simset[sim][filename][item],filepath=file_path)
```

Kuvio 5: ”Clonesimulations”-ohjelma

“parameter_table” (Kuvio 6) luo ajettavaksi sopivan taulukon käyttäjän antamasta parametritiedostosta. “filename”-muuttuja on käyttäjän antama tekstitiedosto, jossa jokainen parametri on erotettu omalle linjalleen, sen kategorian nimi ja arvot erotettu kaksoispisteillä. Parametrin nimi tulee olla aina ensimmäinen argumentti rivillä. Ohjelman tuloste on taulukko kaikista mahdollisista parametriyhdistelmistä.

```
Examples
--------
Example file format python/pencil/pipelines/example_filename.txt
>>> param_table = pc.pipelines.parameter_table("example_filename.txt")
>>> param_table.columns
Index(['run.in/viscosity_run_pars/nu', 'run.in/viscosity_run_pars/ivisc',
       'run.in/viscosity_run_pars/nu_hyper3', 'cparam.local/nxgrid',
       'run.in/entropy_run_pars/chi_hyper3',
       'run.in/entropy_run_pars/iheatcond'],
      dtype='object')
"""
#Extract list of parameters from userfile
lines = open(filename).readlines()
if not quiet:
    print(lines)
#Pipe lines into dictionary
inputs = dict()
for line in lines:
    line = line.strip('\n').replace(' ','')
    inputs[line.split(':')[0]] = list(line.split(':')[1:])
#Extract column headers from inputs
columns = list()
for key in inputs.keys():
    columns.append(key)
#List parameter options for each column
newlist = list()
for key in inputs.keys():
    newlist.append(inputs[key])
#Create data matrix of parameter combinations
data = []
for i in product(*newlist):
    if not quiet:
        print(i)
    data.append(i)
params = pd.DataFrame(data=data,columns=columns)
#Create column indicating whether to compile each clone simulation
params.insert(6, 'compile', False, allow_duplicates=True)
return params
```

Kuvio 6: “parameter_table”-ohjelma

“trim_table”-ohjelma tarvittaessa tiivistää taulukkoa tai lisää siihen rivejä tai sarakkeita. Tätä ohjelmaa tulee kutsua täyden taulukon luomisen jälkeen (parameter_table) ja ennen kuin lopullinen simulaatio luodaan “make_sims_dict”-ohjelmassa.

“make_sims_dict” (Kuvio 7) saa arvot pandas-taulukosta, luodakseen sisäkkäisen “simset”-hakemiston, jota sitten “Clonesimulations” käyttää luodakseen lopullisen simulaatiomatriisin. “params”-muuttuja lukee parametrilistan käyttäjätiedoston rivien ensimmäisistä arvoista.

```
Examples
--------
>>> simset = pc.pipelines.make_sims_dict(params)
>>> simset['nu1e-3nu_hyper31e-9nxgrid400chi_hyper31e-9']
{'run.in'      :
    {'viscosity_run_pars':
        {'nu'        : ' 1e-3',
         'ivisc'     : " ['nu-shock', 'nu-const', 'nu-hyper3']",
         'nu_hyper3' : ' 1e-9'},
     'entropy_run_pars'  :
        {'chi_hyper3': '  1e-9', 'iheatcond': " 'hyper3'"}},
 'cparam.local':
        {'nxgrid'    : ' 400'}}
"""
simset = dict()
#Create nested dictionary of simulation values
for index, row in params.iterrows():
    param_nested_dict = FlatDict(dict(row), delimiter='/').as_dict()
    simkey = ""
    if not quiet:
        print(param_nested_dict)
        print(row.keys)
    for value,key in zip(row,row.keys()):
        #Only 'compile' uses bool. Arguments otherwise are string.
        if not type(value) == bool:
            if not "'" in value:
                simkey = simkey+"{}{}".format(key.split('/')[-1],value.strip(' '))
    simset[simkey] = param_nested_dict
return simset
```

Kuvio 7: “make_sims_dict”-ohjelma

5.5 Sovelluksen testaus

Sovelluskehityksen aikana testaus oli keskeisessä roolissa. Aloitimme testaamaan sovellusta vaihtamalla aluksi vain yhtä parametriä, ja kun se toimi, testaamalla esimerkiksi kokonaislukujen sijasta merkkijonojen lisäämistä. Vaihdoimme myös tiedostoa simulaation sisällä, mitä muokkasimme. Joskus kokonaisluvuista tuli merkkijonoja, ongelma, jonka kuitenkin lopulta ratkaisimme. Muita ongelmia tuottivat myös erityisesti yhteensopivuus olemassa olevan Pencil Code –koodin kanssa.

Lopullista sovellusta testattiin toimivasti syöttämällä ohjelmalle tekstitiedosto, joka sisälsi kuusi eri parametriä, joista jokaisella oli kolmesta viiteen eri arvoa. Onnistunut testaus loi yhdistelmät simulaatioista, jotka sitten ajettiin. Kyseinen tekstitiedosto (Kuvio 8) liitettiin myös osaksi dokumentaatiota esimerkkinä, miten käyttäjän tulisi muotoilla syötettävä parametritiedosto.

```
+ run.in/viscosity_run_pars/nu: 1e-3: 1e-4: 1e-5
+ run.in/viscosity_run_pars/ivisc: ['nu-shock', 'nu-const', 'nu-hyper3']:  ['nu-shock', 'nu-hyper3']
+ run.in/viscosity_run_pars/nu_hyper3: 1e-9: 1e-10: 1e-11
+ cparam.local/nxgrid: 400: 800
+ run.in/entropy_run_pars/chi_hyper3:  1e-9:  1e-10: 1e-11
+ run.in/entropy_run_pars/iheatcond: 'hyper3'
```

Kuvio 8: Esimerkkitiedosto

5.6 Sovelluksen dokumentointi ja julkaisu

Sovellusta ei dokumentoitu kuin vasta viime hetkillä sovelluksen kehityskaaressa. Lähdekoodin kommentointi kehityksen aikana oli lähinnä suttupaperia muistuttaakseen tekijää siitä, mitä koodirivi teki, tai että mitä tähän kohtaan ohjelmaa tulisi lisätä. Kuitenkin lopussa sovelluksen dokumentoitiin ja julkaisemiseen Githubissa käytettiin yksi kokonainen päivä.

Dokumentoinnissa muotoilimme ohjelmakoodia Pencil Coden hyvän dokumentaation ohjeen (Heinemann, 2015.**)** mukaan. Jokaiseen ohjelmaan lisättiin kuvaus siitä mikä sen tarkoitus oli, mitä jokainen muuttuja teki, ja mikä sen tulosteen tuli olla. Annoimme myös käyttäjille myös esimerkkejä siitä, miten muuttujat toimivat. Koodin sisälle tuli lyhyitä kommenttirivejä, selittäen miten se toimii.

Julkaisuun ja versionhallintaan käytimme Git-versionhallintajärjestelmää, josta siirsimme valmiit työkalut osaksi The Pencil Code –projektin Github-repositoriota. Git on hajautettu versionhallintajärjestelmä, jossa käyttäjillä on paikallinen kopio. Käyttäjät täten voivat tehdä muutoksia omaan kopioonsa, jotka he sitten halutessaan voivat liittää projektin pääkehityshaaraan. Järjestelmän idea on, että projekteja voidaan kehittää riippumatta toisistaan ja muutokset eivät suoraan laita vakaata kehityshaaraa epätasapainoon, jonka tarkoitus on olla toimiva aina. Kehitystuote voidaan kehittää sivussa ja toimiessaan liittää osaksi vakaata lopputuotetta. Github on verkossa oleva Git-palvelu, joka komentorivisovelluksen sijasta tarjoaa graafisen käyttöliittymän ja käyttäjähallinnan, jotka tekevät projektien järjestelemisen ja keskittämisen helppokäyttöiseksi ja suoraviivaiseksi. (Karhusaari, 2022.)

6 Yhteenveto ja jatkokehitys

Lopputuote todettiin toimivaksi kehitysryhmämme toimesta ja se jaettiin käytettäviksi projektin käyttäjille Githubissa. Itse olin tyytyväinen lopputulokseen, ottaen huomioon kokemukseni työstä ohjelmoijana ja käytetyistä teknologioista, ja kuinka meidän tuli muutamaan otteeseen jättää asioita pois projektista tai vaihtaa lähestymistapaamme. Oloni oli jopa hieman

yllättynyt, kun palaset loksahtivat niin hyvin paikalleen, tavalla, joka täytti toimeksiantajan odotukset.

Työharjoitteluni loppuessa juuri työkalun valmistuessa, en saanut tietää tarkalleen, miten se otettiin vastaan. En kuullut takaisin toimeksiantajalta kuin vasta keväällä 2022, kun he tarjosivat minulle jälleen työtä kesäksi projektin jatkokehityksen parissa. Silloinkaan minulle ei kerrottu, oliko tuote napakymppi vai floppi vai jotain siltä väliltä. Oletan kuitenkin sen, että siinä on nähty potentiaalia, jonka takia sen kehitystä jatketaan.

Yksi näistä potentiaalisista jatkokehityssuunnista voisi olla joku niistä alkuperäiseen määrittelyyn kuuluneista ominaisuuksista: datan postprosessointi ja visualisointi, diagnostiikat ja MPI-toimivuus. Varsinkin kahden ensin mainitun tuli olla osa tätä alkuperäistä työkalua, mutta ne jätettiin pois.

7 Oman oppimisen arviointi

Työharjoittelu The Pencil Code –projektin parissa oli ensimmäinen työpaikkani, missä työskentelin ohjelmoijana. Aiemmat opintoni antoivat minulle kohtalaisen hyvän pohjan ja ymmärryksen siitä, mitä ohjelmointi ja projektityöskentely ovat, mutta kokemukseni siitä jäivät ennen työharjoittelua vielä opintojaksoilla työstettyihin pieniin töihin.

Projektikehityksen aikana ja sen jälkeen opin kuitenkin asioita, jotka tekevät minut itsevarmemmaksi oppimiskyvystäni ja tulevaisuudestani ohjelmointitöissä. Vaikka emme sinänsä käyttäneet mitään klassisia kehitysmalleja kuten vesiputousmallia työssämme, opin silti, miten työskennellä pitkäkestoisesti toisten ohjelmoijien rinnalla keskisuuren mittakaavan projektia työstäen. Puhumattakaan uusien teknologioiden ja kehitysympäristöjen omaksumisesta, jossa koin oppineeni kuudessa kuukaudessa enemmän kuin vuosien aikana ennen sitä.

Projektikehityksessä kohtaamani ongelmat juuri useiden eri teknologioiden parissa pakottivat minut oppimaan paljon uutta, ja ajattelemaan asioita erilaisista näkökulmista. Tieto projektin oikeasta tarpeesta antoi myös aivan uudenlaista motivaatiota työskentelyyn ja ylpeyttä esitellä sitä niille, jotka sitä tarvitsivat. Tämän kautta myös esiintymistaitoni paranivat ja sain uutta kokemusta siitä, miten pitää demonstraatioita ja esitelmiä.

Jotain mitä nyt tekisin toisin olisi kenties yrittää etsiä enemmän ratkaisuja minulle annetuiden ohjenuorien ulkopuolelta. Toimin pääosin yhdessä päätettyjen raamien sisällä, käyttäen teknologioita, joita päätimme yhdessä käyttää, sen sijaan että olisin itse yrittänyt etsiä enemmän uusia ratkaisuja niiden ulkopuolelta. Näin olisin ehkä voinut luoda tehokkaammin

uutta koodia, enkä pyöriä esimerkiksi kauaa jonkun Python-moduulin parissa, jota emme lopulta kuitenkaan päättäneet käyttää projektissamme.

Lähteet

Painetut

Brandenburg A., Dobler W. 2002. Hydromagnetic turbulence in computer simulations. Alankomaat: Computer Physics Communications.

Sähköiset

Aalto-yliopisto 2022. Tietotekniikan laitos. Viitattu 6.6.2022. https://research.aalto.fi/fi/organisations/department-of-computer-science

Aalto-yliopisto 2021. Astroinformatics. Viitattu 6.6.2022. https://www.aalto.fi/en/department-of-computer-science/astroinformatics

The Visual Room 2022. 4. Sod's Test Problems: The Shock Tube Problem. Viitattu 6.6.2022. http://www.thevisualroom.com/sods_test_problem.html

Red Hat 2018. Understanding Linux. Viitattu 6.6.2022. https://www.redhat.com/en/topics/linux

CSC – Tietotekniikankeskus 2022. Tietoa meistä. Viitattu 6.6.2022. https://www.csc.fi/tietoa-meista

Zinoune, M. 2022. Why do super computers use Linux? Viitattu 6.6.2022. https://www.unixmen.com/why-do-super-computers-use-linux/

Jupyter Notebook 2015. What is the Jupyter Notebook? Viitattu 6.6.2022. https://jupyter-notebook-beginner-guide.readthedocs.io/en/latest/what_is_jupyter.html

Python Suomi ry 2022. Yleistä. Viitattu 6.6.2022. http://python.fi/yhdistys/

MOOC.org 2021. What is Python used for? Viitattu 6.6.2022. https://www.mooc.org/blog/what-is-python-used-for

NumPy 2022. Learn. Viitattu 6.6.2022. https://numpy.org/learn/

pandas 2022. About pandas. Viitattu 6.6.2022. https://pandas.pydata.org/about/

Turun yliopisto 2015. Esimerkki: Extreme Programming (XP). Viitattu 6.6.2022. https://tt.utu.fi/embedded_kasikirja/1/3/index.html

Brandenburg, A., Dobler, W. 2022. The Pencil Code: A High-Order MPI code for MHD Turbulence. Viitattu 6.6.2022. https://github.com/pencil-code/website/raw/master/doc/manual.pdf

Losada, I. R., Lambrechts, M., Cole, E., Bourdin, P. 2021. Pencil Code: Quick Start guide. Viitattu 6.6.2022. https://github.com/pencil-code/website/raw/master/doc/quick_start.pdf

Open MPI 2022. Frequently Asked Questions. Viitattu 6.6.2022. https://www.open-mpi.org/faq/

Heinemann, T. 2015. Coding style checklist. Viitattu 6.6.2022. https://github.com/pencil-code/pencil-code/wiki/CodingStyle

Karhusaari, M. 2022. Osa 2 – Versionhallinta: Git ja Github. Viitattu 6.6.2022. https://tkt-lapio.github.io/git/

Kuviot

Kuvio 1: Yksinkertainen harjoitus graafisen esityksen luomisesta 8
Kuvio 2: Vi-kehitysympäristö..... 10
Kuvio 3: Jupyter Notebook -kehitysympäristö 11
Kuvio 4: Sod shocktube –ohjelman parametrinäkymä..... 15
Kuvio 5: ”Clonesimulations”-ohjelma 18
Kuvio 6: “parameter_table”-ohjelma 19
Kuvio 7: “make_sims_dict”-ohjelma..... 20
Kuvio 8: Esimerkkitiedosto 21