Meanderoivan jokiuoman migraation mallintaminen

Oona Oksanen

Pro gradu -tutkielma
Tammikuu 2025

Tarkastajat:
Prof. Marko Mäkelä
Dos. Elina Kasvi

MATEMATIIKAN JA TILASTOTIETEEN LAITOS

TURUN YLIOPISTO
Matematiikan ja tilastotieteen laitos

Oona Oksanen: Meanderoivan jokiuoman migraation mallintaminen
Pro gradu -tutkielma, 61 s., 17 liites.
Sovellettu matematiikka
Tammikuu 2025

Meanderoivat jokiuomat ovat mutkittelevia jokia, joiden muoto ja sijainti muuttuvat ajan myötä luonnon monimutkaisten prosessien, kuten veden virtauksen, sedimentaation ja eroosion vaikutuksesta. Nämä prosessit ilmenevät joen profiilissa ja sen geometrisissa ominaisuuksissa. Jokiuomien piirteiden analysointi syventää ymmärrystä geomorfologisten muodostumien synnystä ja kehityksestä. Matemaattiset mallit tarjoavat keinoja hahmottaa näitä muutoksia ja ennustaa jokiuomien tulevaa kehitystä.

Tässä tutkielmassa kehitetään mallinnusmenetelmä, jonka avulla voidaan ennustaa meanderoivien jokiuomien migraatiota eli siirtymäliikettä tulevaisuudessa. Menetelmän toiminta perustuu ChaRMigS-menetelmään, joka simuloi jokiuoman historiallisia reittejä ja tarjoaa skenaarioita uoman muutoksista. Kehitettävässä menetelmässä migraatiosuunnat käännetään päinvastaisiksi ChaRMigS-menetelmään nähden ja algoritmiin lisätään ehto uusien juoluoiden muodostumiselle. Lisäksi uutena osana otetaan käyttöön sallittu alue, jonka sisällä migraatio voi tapahtua.

Kehitettyä menetelmää sovelletaan Oulankajoen ja Pulmankijoen jokiuomien tulevan migraation mallintamiseen. Lopputuloksena saadaan simulaatioita, jotka esittävät mahdollisia kehityspolkuja jokiuomien muutoksille tulevaisuudessa. Tavoitteena on tarjota työkalu, joka auttaa ymmärtämään jokiuomien pitkän aikavälin kehitystä.

Asiasanat: Meanderoiva jokiuoma, matemaattinen mallinnus, migraation mallinnus, simulaatio, tulevaisuuden ennuste

UNIVERSITY OF TURKU
Department of Mathematics and Statistics

Oona Oksanen: Mathematical modeling of meandering river channel migration
Master's Thesis, 61 pp., 17 app.
Applied Mathematics
January 2025

Meandering river channels are sinuous rivers whose shape and location evolve over time due to complex natural processes, such as water flow, sedimentation, and erosion. These processes are reflected in the river's profile and its geometric features. Analyzing these features enhances the understanding of the formation and evolution of geomorphological structures. Mathematical models provide tools to study changes in river channels and predict their future development.

This thesis develops a modeling method to predict the future evolution of meandering river channels. The method is based on the ChaRMigS approach that simulates the historical trajectories of river channels and generates scenarios of channel changes. In the method introduced in this thesis, the migration directions are reversed compared to the ChaRMigS approach, and a condition for the formation of new oxbow lakes is added to the algorithm. Additionally, a feasible region within which migration can occur is introduced as a new component.

Here, the method is applied to model the future migration of the Oulankajoki and Pulmankijoki river channels. As a result, the method produces simulations that represent possible development pathways for river channel evolutions. The aim is to provide a tool that helps understand the long-term evolution of river channels.

Keywords: Meandering river channel, mathematical modeling, migration modeling, simulation, future prediction

Sisällys

1 Johdanto

Tässä Pro gradu -tutkielmassa sovelletaan Parquer ym. [21] kehittämää ChaRMigS *(The Channel Reverse Migration Simulation)* -menetelmää, joka mallintaa meanderoivan jokiuoman migraatiota eli uoman siirtymäliikettä. ChaRMigS-menetelmä simuloi joen mahdollisia reittejä historiassa ja luo skenaarioita siitä, miten jokiuoma on saattanut muuttua ajan kuluessa. Tämän tutkielman tavoitteena on kehittää ChaRMigS-menetelmän pohjalta uusi menetelmä, joka mahdollistaa meanderoivan jokiuoman tulevan kehityksen mallintamisen ja tarjoaa ennusteita sen mahdollisista muutoksista tulevaisuudessa.

Meanderoivalla jokiuomalla tarkoitetaan kaareilevaa jokiuomaa. Tällaiset uomat tarvitsevat muodostuakseen paksun maakerroksen ja hienojakoista hyvin lajittunutta maaperää. Meanderoituminen edellyttää myös melko tasaista topografiaa sekä matalia virtausnopeuksia. Jokiuoman kaarteet muodostuvat, kun ulkokaarteiden voimakkaampi virtaus aiheuttaa eroosiota, samalla kun sisäkaarteiden hitaampi virtaus kasaa maa-ainesta. Tämä prosessi muokkaa jokiuomaa ajan myötä ja johtaa uoman migraatioon. Joskus vierekkäiset kaarteet saattavat kuroutua kiinni toisiinsa, minkä seurauksena uoma oikenee ja väliin jäävän osa irtoaa muodostaen lammen tai järven, jota kutsutaan juoluaksi. Tämä muokkaa jokiuomaa ja sen ympäristöä entisestään [9].

ChaRMigS-menetelmä mallintaa migraatioprosessia meanderoivalle jokiuomalle ajassa taaksepäin. Mallinnusmenetelmä perustuu jokiuoman keskiviivan geometriaan, ja lisäksi kaarteiden migraatiossa käytetään stokastisuutta jäljittelemään reaalimaailman muuttuvia olosuhteita. Menetelmällä voidaan mallintaa meanderikaarteiden laajentumista, siirtymistä sekä juoluoiden liittämistä takaisin osaksi jokiuomaa. Menetelmän tavoitteena on hahmottamaan jokiuoman mahdollisia kehityspolkuja historiassa.

ChaRMigS-menetelmää on käytetty mallintamaan eri jokialueilta, kuten Suomessa sijaitsevaa Oulankajokea, Thaimaassa sijaitsevaa Chao Phraya -jokea ja Venäjällä sijaitsevaa Tangnara-jokea. Simulaatioilla on saatu uskottavia tuloksia jokiuomien historiasta sekä onnistuttu integroimaan joitain juoluoita takaisin osaksi joen reittiä [21, 22].

Tässä tutkielmassa tavoitteena on kehittää mallinnusmenetelmä, jolla voidaan ennustaa jokiuoman tulevaa kehitystä. Kehitetty menetelmä pohjautuu ChaRMigS-algoritmin toimintaan ja hyödyntää samaa migraatiomallia uoman siirtymien simulointiin. Tulevaisuutta mallinnettaessa liikkumissuunnat määritellään päinvastaisiksi alkuperäiseen CharRMigS-menetelmään verrattuna ja määritellään ehto uusien juoluoiden muodostumiselle. Lisäksi uutena osana simulaatioon lisätään sallittu alue,

jonka sisällä joen migraatio saa tapahtua. Näin mallinnusmenetelmän tuloksista saadaan realistisempia.

Kehitettyä menetelmää sovelletaan kahden Suomen alueella virtaavan meanderoivan joen, Oulankajoen ja Pulmankijoen, tulevan migraation ennustamiseen. Mallinnuksessa tarkastellaan noin 25 kilometrin pituista osuutta Oulankajoesta ja noin 9 kilometrin pituista osuutta Pulmankijoesta. Sallittuna alueena käytetään Maaperäaineiston [27] avulla rajattuja jokilaakson alueita, missä uomien migraatio on mahdollista. Lopputuloksena simulaatiot tuottavat mahdollisia reittejä, joita jokiuomat voivat kulkea tulevaisuudessa.

Tämän tutkielman laatimisessa olen käyttänyt ChatGPT-tekoälysovellusta [20] kielenhuoltoon, jotta teksti olisi selkeämpää ja helpommin ymmärrettävää. Lisäksi olen hyödyntänyt ChatGPT:tä python-koodin optimointiin ja ohjelmointitekniikoiden oppimiseen erityisesti paikkatietoaineistojen käsittelyssä.

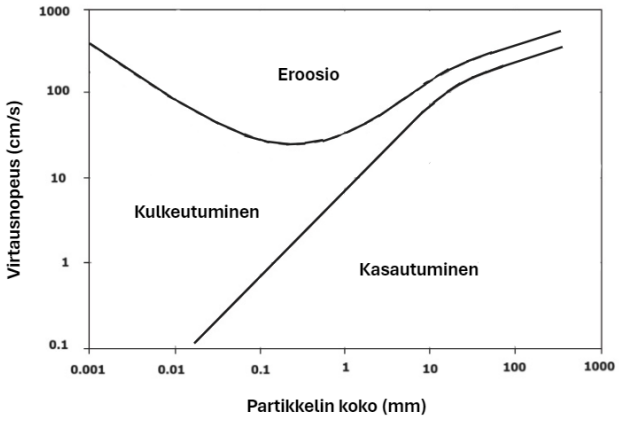
2 Meanderoiva jokiuoma ja sen geometria

Tässä luvussa käsitellään meanderoivan jokiuoman teoriaa ja geometriaa. Luku perustuu kirjoihin [3, 9].

2.1 Uoman geomorfologia ja fluviaaliset prosessit

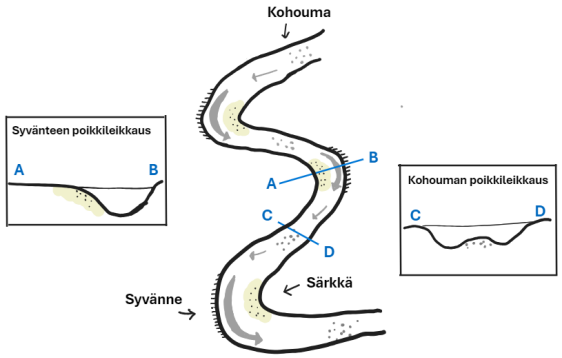
Jokiprofiilit voidaan luokitella neljään eri tyyppiin: suoraan, meanderoivaan, palmikoivaan sekä jälleenyhdistyvään eli anastomoivaan. Jokiuoman geomorfologia, eli sen muoto, riippuu veden virtauksesta, maaperästä, kasvillisuudesta ja topografiasta. Meanderoivien jokien kaareileva muoto johtuu virtaavan veden aikaansaamasta eroosiosta sekä sedimenttien kuljetuksesta ja kasaantumisesta.

Kuluminen, kuljetus ja kasautuminen ovat veden virtauksen aiheuttamia fluviaalisia prosesseja, jotka ovat keskeisiä tekijöitä joen geomorfologian kehityksessä. Kulumisessa joen uoma syventyy tai leventyy, kun sedimentti irtoaa ja lähtee kulkeutumaan virtaavan veden mukana. Kasaantumisessa taas sedimentti kerääntyy muodostaen särkkiä ja kohoumia, jotka madaltavat tai kaventavat jokiuomaa. Sedimentin kulkeutumista voidaan havainnollistaa Hjulströmin [8] diagrammilla, joka kuvaa raekoon ja virtausnopeuden suhdetta kulkeutumiseen, kulumiseen ja kasautumiseen (kuva 1).



Kuva 1: Hjulströmin diagrammi kuvaa partikkelikoon ja virtausnopeuden välistä suhdetta havainnollistaen sedimentin kulkeutumisen, kulumisen ja kasaantumisen prosesseja jokiuomassa (Hjulström [8] mukaillen).

Meanderoivalle joelle tunnusomaisia piirteitä ovat kulumis- ja kasaantumisprosessissa syntyneet meanderikaarteet, särkät, syvänteet ja kohoumat (kuva 2). Syvänteet ja särkät sijaitsevat kaarteissa ja kohoumat uoman käännekohdissa.

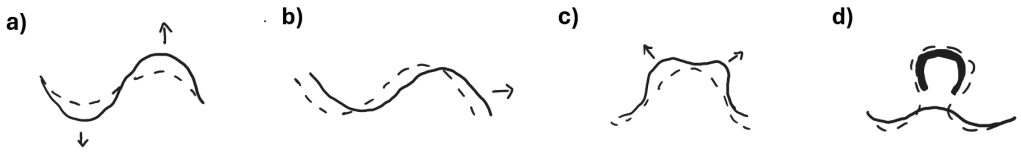


Kuva 2: Syvänteet muodostuvat uoman ulkokaarteisiin, joissa nopea veden virtaus aiheutuu eroosiota. Särkät ja kohoumat puolestaan syntyvät sisäkaarteisiin ja suoriin osuuksiin, joissa virtaus on hitaampaa, mikä mahdollistaa sedimentin kertymisen (Charlton [3] mukaillen).

Päävirtauksen lisäksi meanderoivissa joissa vaikuttaa sekundaarivirtaus. Sekundaarivirtaus on meanderikaarteisiin muodostuva päävirtausta vasten kohtisuorassa oleva virtaus joen pinnan ja pohjan välillä. Sekundaarivirtaus ohjaa myös sedimentin kulkeutumista ja muokkaa uomaa erityisesti kaarteissa.

2.2 Migraatio

Meanderoiva jokiuoma muuttaa muotoaan ajan myötä ja tätä prosessia kutsutaan migraatioksi. Migraatio voi olla sivuttaissuuntaista (kuva 3a), jolloin kaarteet levenevät ja niiden kokonaispituus kasvaa. Virtaussuuntaan tapahtuva migraatio (kuva 3b) taas liikuttaa joen kaarteita eteenpäin. Meanderikaarteet saattavat alkaa myös jakautumaan kahteen suuntaan (kuva 3c). Lisäksi juoluoiden syntyminen on yksi joen migraatioon liittyvä ilmiö. Meanderikaarteiden liikkuessa tarpeeksi lähelle toisiaan, joen virtaus saattaa oikaista reitin, jolloin väliin jäänyt mutka jää erilleen jokiuomasta muodostaen juoluan (kuva 3d). Nämä migraatioprosessit tapahtuvat usein rinnakkain, mikä johtaa jokiuoman moniulotteisiin muutoksiin.

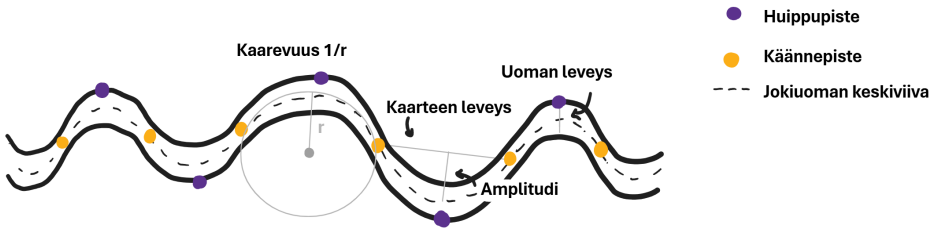


Kuva 3: Meanderikaarteiden migraation suunnat. a) Sivuttainen migraatio laajentaa kaarteita. b) Virtaussuuntaan tapahtuva migraatio kuljettaa kaarteita uoman pituussuunnassa. c) Yksittäinen kaarre saattaa jakautua eri suuntiin. d) Juolua muodostuu, kun vierekkäiset kaarteet yhdistyvät ja joen reitti suoristuu (Charlton [3] mukaillen).

Jokiuomien migraatiota voidaan seurata muun muassa satelliitti- ja ilmakuvia tarkastelemalla [2]. Kuvista voidaan mitata uoman siirtymistä vuosien aikana ja tarkastella kuinka paljon ja miten jokiuoma on muuttunut. Pitkän aikavälin muutosten arviointiin voidaan käyttää myös historiallisia karttoja [21].

2.3 Geometrisiä ominaisuuksia

Meanderoivan joen profiilille voidaan laskea geometrisia ominaisuuksia, joiden avulla jokiuomaa ja sen muutoksia voidaan analysoida (kuva 4). Jokiuomalle voidaan laskea *pituus* uoman keskiviivaa pitkin, sekä *leveys* kahden vastakkaisen reunapisteen etäisyytenä. Yksittäisten kaarteiden kokoa voidaan tarkastella laskemalla *kaarteen leveys* sekä *amplitudi* eli laajuus. Meanderikaarteiden *kaarevuus* voidaan määrittää sovittamalla kaarteeseen ympyrä, jonka säteen käänteisluku $\frac{1}{r}$ ilmaisee kaarevuutta. Vaihtoehtoisesti jokiuoman keskiviivalle voidaan laskea paikallisia kaarevuusarvoja käyrän kaarevuuden mukaisesti. *Käännepisteet* kertovat jokiuoman kohdat, joissa kuperuussuunta vaihtuu konveksista konkaaviin tai toisinpäin, ja *huippupisteet* kertovat kaarteiden äärikohdat.

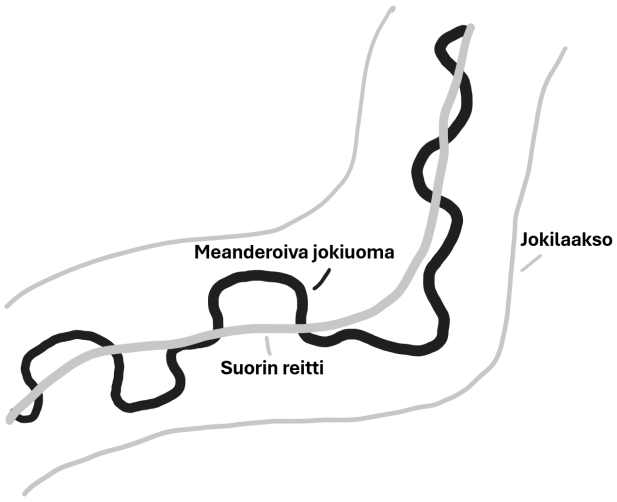


Kuva 4: Joen profiilin geometrisia ominaisuuksia, joiden avulla jokiuomaa voidaan matemaattisesti analysoida.

Jokiuoman kaareilevuuden astetta mitataan *sinuositeetilla* S (kuva 5). Sinuositeetti saadaan vertaamalla joen todellista pituutta suorimpaan mahdolliseen reittiin,

jonka jokiuoma voisi kulkea sen alku- ja loppupisteen välillä

$$S = \frac{\text{pituus virtaa pitkin}}{\text{suorimman mahdollisen reitin pituus}}.$$



Kuva 5: Jokiuoman sinuositeetti saadaan laskemalla jokiuoman todellinen pituus (musta) ja jakamalla se suorimman mahdollisen reitin pituudella (harmaa). Tämä kuvaa kaareilevuuden astetta.

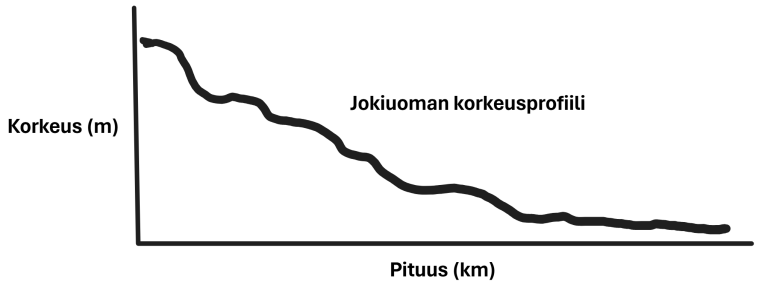
Sinuositeetin S avulla voidaan luokitella joen mutkittelun aste ja arvioida, kuinka paljon se poikkeaa suorasta reitistä. Taulukko 1 esittää sinuositeetin perusteella tehtävän luokittelun.

Taulukko 1: Jokiuoman sinuositeetin luokittelu.

$S < 1.1$	Suora uoma
$1.1 < S < 1.5$	Loivasti meanderoiva uoma
$S \geq 1.5$	Selkeästi meanderoiva uoma

Meanderoivat joet esiintyvät tyypillisesti laaksoissa, joissa kaltevuudet ovat alhaisia. Jokiuoman *kaltevuus* kuvaa kuinka jyrkästi uoma laskee alku- ja loppupisteen välillä (kuva 6). Kaltevuus määritellään korkeuseron suhteena uoman kulkeman matkan pituuteen

$$\text{Kaltevuus} = \frac{\text{korkeusero}}{\text{alku- ja loppupisteen välinen etäisyys}}.$$

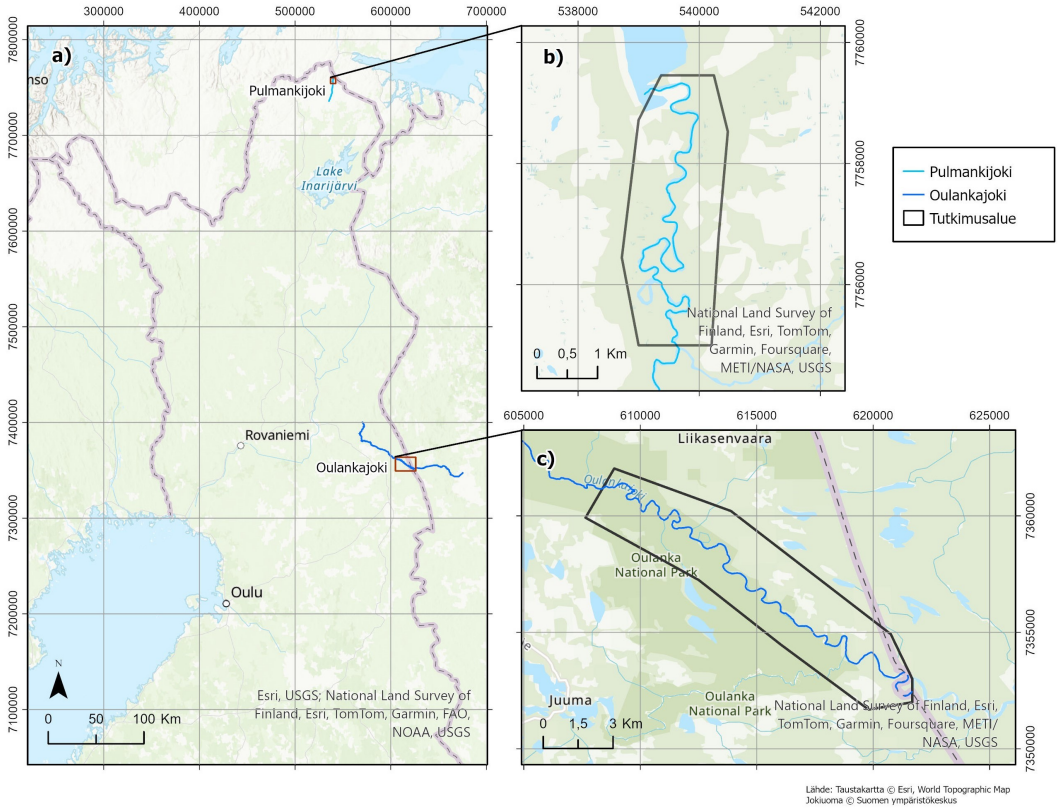


Kuva 6: Jokiuoman kaltevuus saadaan jakamalla uoman korkeusmuutos sen kulkeman matkan pituudella. Kaltevuus ilmaistaan keskimääräisenä korkeuden muutoksena kilometriä kohden ($\frac{m}{km}$).

Meanderoivan jokiuoman kehitys perustuu luonnon monimutkaisiin prosesseihin. Nämä heijastuvat joen profiiliin ja sen geometrisiin ominaisuuksiin. Ominaisuuksien analysointi syventää ymmärrystä geomorfologisten piirteiden muodostumisesta ja tarjoaa mahdollisuuksia ennustaa uomien kehitystä.

3 Aineisto ja tutkimusalueet

Tulevaisuudelle kehitettyä mallinnusmenetelmää havainnollistetaan tutkielman lopussa kahdelle Suomessa virtaavalle meanderoivalle joelle, Oulankajoelle ja Pulmankijoelle (kuva 7a). Oulankajoki sijaitsee Oulangan kansallispuistossa, jossa se virtaa Sallasta kohti Venäjällä sijaitsevaa Paanajärveä. Oulankajoen pituus Suomen alueella on noin 110 km. Mallinnuksessa keskitytään noin 24.5 kilometrin mittaiseen osuuteen, joka ulottuu Oulangan kansallispuiston alueelta Venäjän rajalle (kuva 7c). Pulmankijoki taas sijaitsee Utsjoella Kaldoaivin erämaa-alueella, jossa se virtaa kohti Suomen ja Norjan rajalla olevaa Pulmankijärveä. Pulmankijoen pituus on noin 35 km, josta mallinnukseen on otettu noin 9.5 km mittainen osuus (kuva 7b).



Kuva 7: Tutkimusalueet. a) Tutkimusalueiden sijainnit. b) Pulmankijoen tutkimusalueena on noin 9.5 km pituinen jokiosuus. c) Oulankajoen tutkimusalue kattaa noin 24.5 km pitkän osuuden jokiuomasta Oulangan kansallispuiston alueelta.

Oulankajoen laakso on muotoutunut noin puolitoista miljardia vuotta sitten tapahtuneen maankuoren halkeilun seurauksena, ja viimeisimmän jääkauden suuret jäämassat ovat muokanneet sitä nykymuotoon. Jäätikköjokien kuljettama hiekka-

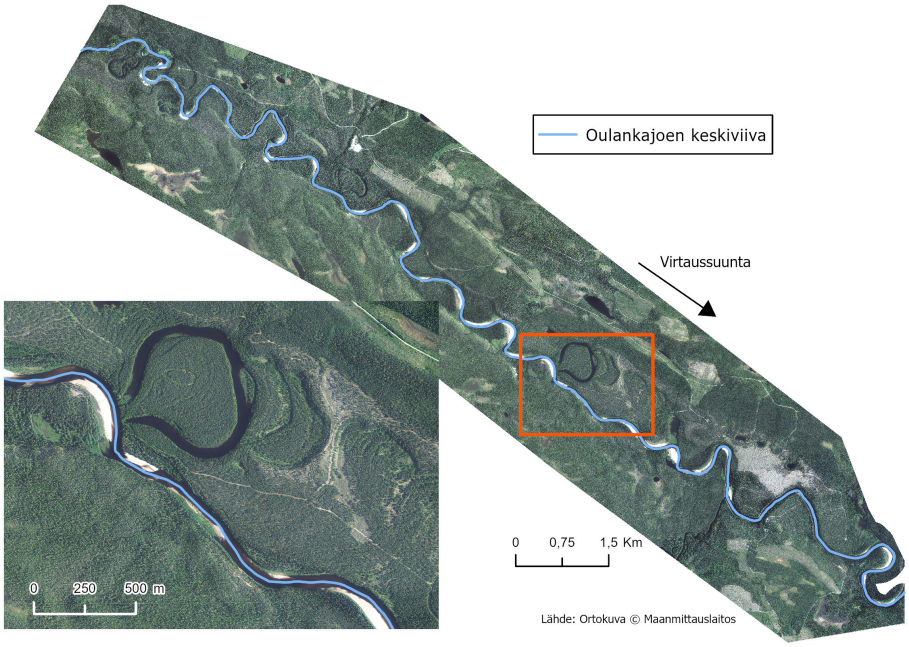
ja sora-aines on täyttänyt jokilaakson, jonka läpi Oulankajoki on alkanut virrata kohti itää. Nykyisin jokiuoma sijaitsee noin 20-35 metrin syvyydessä hiekkakerrostumassa. Jokilaakson kaltevuus on noin 0.3 m/km [24, 13].

Myös Pulmankijoen laakso on muodostunut jäämassojen muokkaamana ja sulamisvesien kuljettaessa maa-ainesta kohti Jäämerta. Virtaus on kuluttanut Pulmankijoen jokiuoman 10–30 metrin syvyyteen maakerrokseen. Pulmankijoen jokilaakson kaltevuuden on mitattu olevan noin 0.7 m/km [18, 19].

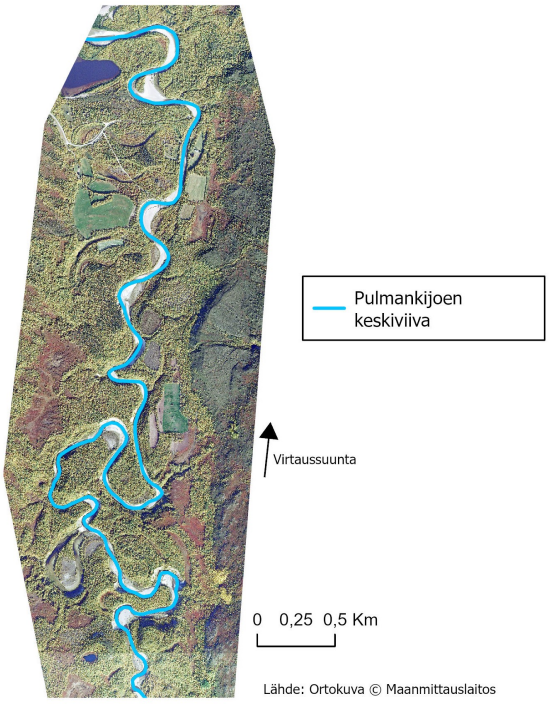
Nykyiset jokiuomat on määritelty mallinnusmenetelmää varten uusimmista Maanmittauslaitoksen Karttapaikka-palvelusta [15] saatavilla olevista ortokuvista. Oulankajoen ortokuva on vuodelta 2022 [29] ja Pulmankijoen vuodelta 2015 [28]. Ortokuvat ovat rasterimuotoa ja jokainen pikseli vastaa $0.5\text{m} \times 0.5\text{m}$ kokoista aluetta. Kuvien avulla jokiuomat on digitoitu käyttäen ArcGIS-paikkatieto-ohjelmaa [4].

Jokiuomat on digitoitu aluksi polygoneiksi. Polygonit sisältävät koordinaattipisteet, jotka määrittävät jokiuoman veden peittämät alueet. Digitointi tehdään manuaalisesti ilmakuvien visuaalisen tulkinnan perusteella. Digitointiprosessissa on käytetty 1:5000 mittakaavaa. Tämän jälkeen polygoneille on luotu keskiviivat käyttämällä ArcGIS-ohjelman automaattista Centerline-työkalua (kuvat 8 ja 9).

Keskiviiva kuvaa yksinkertaisesti joen pääuomaa ja sen avulla voidaan tarkastella jokiuoman ominaisuuksia ja muutoksia siinä. Keskiviiva on vektorimuotoinen aineisto, jota voidaan jatkokäsitellä muun muassa Pythonin geopandas-kirjastolla [6], kuten tässä tutkielmassa on tehty. Jotta jokiuomaa voidaan tutkia samoin kuin Parquer ym. [21] esittävät artikkelissaan, keskiviiva jaetaan myöhemmin vielä tasamittaisiin osaväleihin.

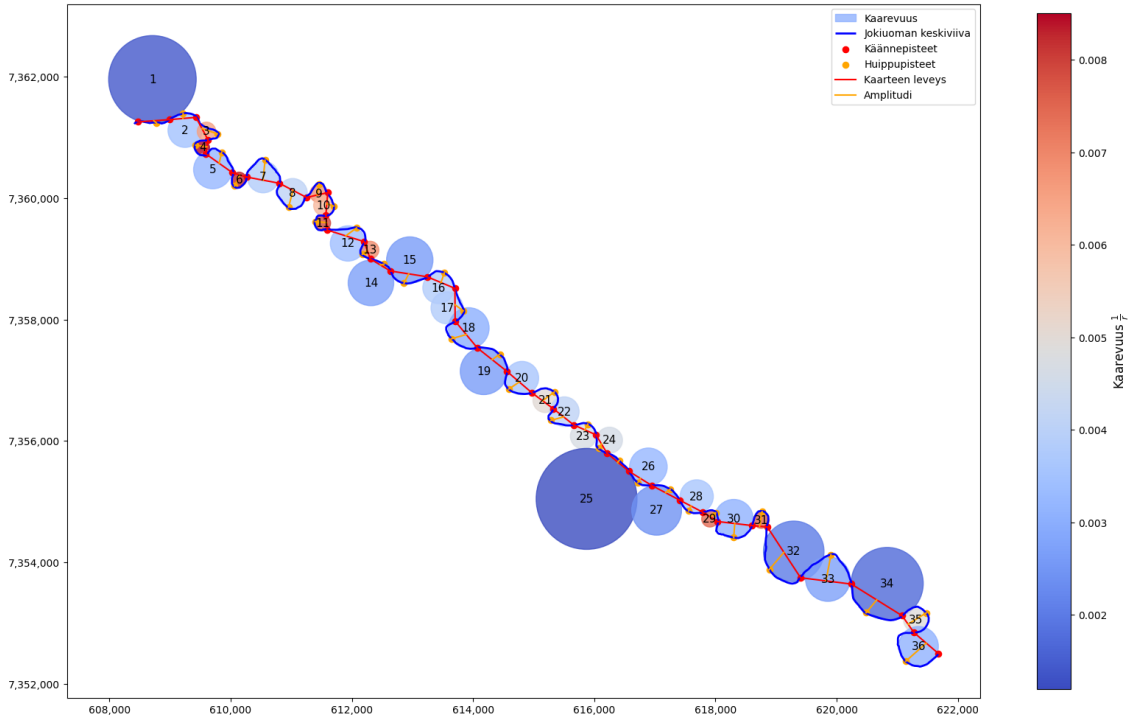


Kuva 8: Oulankajoen keskiviiva digitoituna ortokuvasta.



Kuva 9: Pulmankijoen keskiviiva digitoituna ortokuvasta.

Kuvassa 10 nähdään luvun 2 mukaisia geometrisia ominaisuuksia Oulankajoen tutkimusalueelle. Tulokset on koottuna taulukkoon 2. Sinuositeetiksi Oulankajoen tutkittavalle osuudelle saadaan 1.5, jolloin jokiuoma on taulukon 1 mukaan selkeästi meanderoiva. Sinuositetti on laskettu jakamalla keskiviivan kokonaispituus siihen sovitetun suoran pituudella.

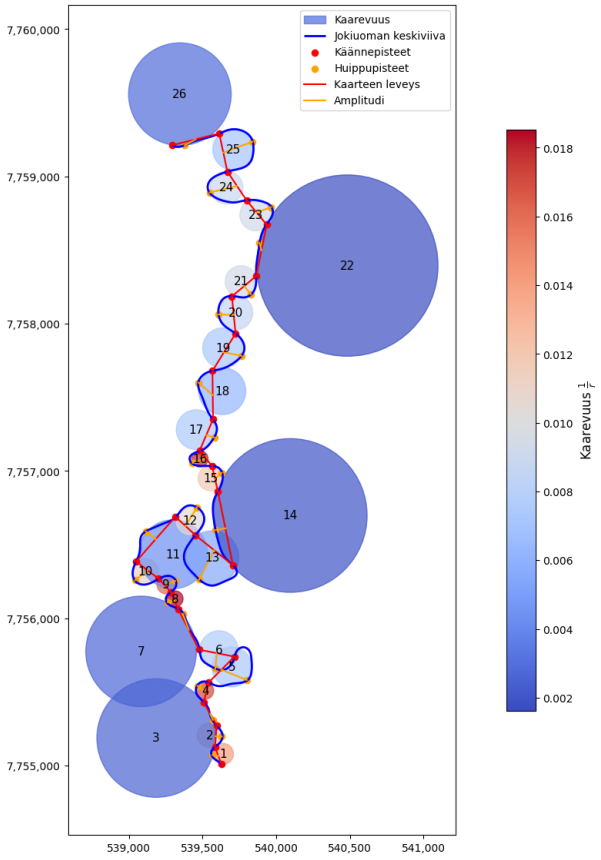


Kuva 10: Oulankajoen kaarteiden geometrisia ominaisuuksia. Kaarteisiin sovitettujen ympyröiden säteen käänteisluku $\frac{1}{r}$ kertoo kaarteen kaarevuudesta. Käännepisteiden välisten suorien (punainen) pituudet kertovat kaarteiden leveydestä, ja etäisyys tästä kaarteen huippupisteeseen kertoo amplitudin (oranssi). Tulokset nähdään taulukosta 2.

Taulukko 2: Oulankajoen kaarteiden tunnuslukuja.

Kaarre	Kaarteen pituus virtaa pitkin (m)	Kaarteen leveys (m)	Amplitudi (m)	Sovitetun ympyrän säde r	Kaarevuus 1/r
1	543.9	533.6	42.4	722.7	0.0014
2	472.1	431.7	80.8	281.8	0.0035
3	652.4	421.2	181.0	153.0	0.0065
4	513.7	234.0	187.7	117.3	0.0085
5	682.4	540.3	174.7	316.5	0.0032
6	518.5	244.8	198.2	120.2	0.0083
7	925.1	546.2	328.8	261.9	0.0038
8	847.3	509.2	282.3	241.5	0.0041
9	520.9	357.5	171.5	146.8	0.0068
10	461.1	372.7	115.1	166.9	0.0060
11	509.4	247.9	187.7	125.9	0.0079
12	785.5	635.0	178.7	290.9	0.0034
13	372.8	302.5	98.1	144.6	0.0069
14	398.9	387.1	43.3	376.2	0.0027
15	712.2	611.9	161.3	382.7	0.0026
16	626.6	498.4	169.3	260.2	0.0038
17	625.5	540.5	135.4	264.3	0.0038
18	848.9	575.3	234.5	336.6	0.0030
19	712.8	619.3	148.5	386.7	0.0026
20	708.0	541.1	202.5	274.9	0.0036
21	706.0	448.3	231.0	201.4	0.0050
22	603.0	424.6	175.2	243.9	0.0041
23	469.5	402.8	107.1	212.2	0.0047
24	383.6	350.4	70.9	215.1	0.0046
25	477.9	470.4	37.6	832.1	0.0012
26	474.1	441.5	78.8	310.0	0.0032
27	556.9	519.9	87.7	411.6	0.0024
28	474.9	429.0	91.7	274.6	0.0036
29	377.5	293.6	103.9	135.4	0.0074
30	842.5	569.3	231.3	319.7	0.0031
31	601.9	255.3	242.2	130.8	0.0076
32	1336.4	1000.4	355.2	499.9	0.0020
33	1260.4	844.7	437.3	368.5	0.0027
34	1187.8	987.6	273.4	595.0	0.0017
35	940.4	339.5	352.4	204.3	0.0049
36	1333.1	534.0	445.7	327.3	0.0031

Kuvassa 11 nähdään Pulmankijoen tutkimusalueen geometrisia ominaisuuksia, joiden tulokset on koottuna taulukkoon 3. Pulmankijoelle sinuositeetiksi saadaan 2.1, jolloin tutkittava osuus uomasta on taulukon 1 mukaan selkeästi meanderoiva.



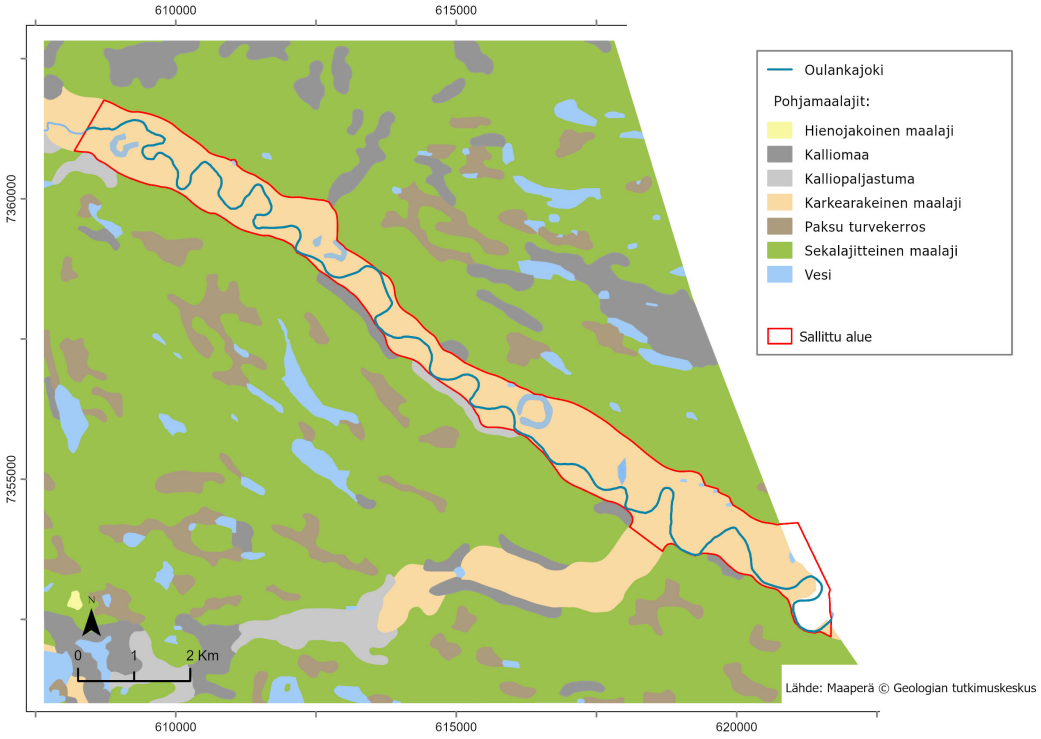
Kuva 11: Pulmankijoen kaarteiden geometrisia ominaisuuksia. Kaarteisiin sovitettujen ympyröiden säteen käänteisluku $\frac{1}{r}$ kertoo kaarteen kaarevuudesta. Käännepisteiden välisten suorien pituudet kertovat kaarteiden leveydestä (punainen), ja etäisyys tästä kaarteen huippupisteeseen kertoo amplitudin (oranssi). Tulokset nähdään taulukosta 3.

Taulukko 3: Pulmankijoen kaarteiden tunnuslukuja.

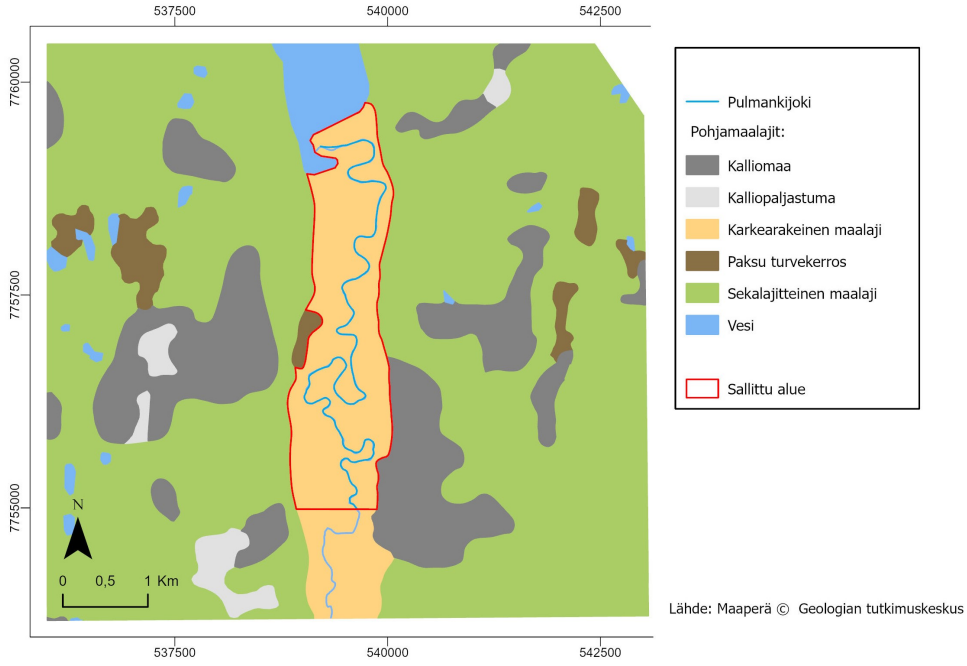
Kaarre	Kaarteen pituus virtaa pitkin (m)	Kaarteen leveys (m)	Amplitudi (m)	Sovitetun ympyrän säde r	Kaarevuus 1/r
1	147.2	123.5	34.7	70.9	0.0141
2	173.5	150.5	40.7	82.8	0.0121
3	178.5	177.7	4.4	402.7	0.0025
4	208.4	137.8	65.9	58.6	0.0171
5	609.0	250.6	171.1	135.8	0.0074
6	353.6	244.1	105.1	130.0	0.0077
7	311.2	307.0	14.9	375.3	0.0027
8	163.6	129.4	46.1	54.0	0.0185
9	208.6	122.1	71.8	62.1	0.0161
10	308.8	193.4	104.6	88.0	0.0114
11	444.2	404.9	83.8	234.1	0.0043
12	396.3	185.2	146.0	96.2	0.0104
13	700.6	325.9	219.1	181.6	0.0055
14	541.4	510.2	67.5	521.9	0.0019
15	216.9	177.6	53.8	84.6	0.0118
16	250.6	134.8	89.1	55.8	0.0179
17	266.5	231.5	56.4	135.1	0.0074
18	400.5	329.2	94.2	160.0	0.0062
19	401.0	297.7	113.5	136.7	0.0073
20	353.5	253.9	102.1	117.8	0.0085
21	265.9	215.6	71.6	107.3	0.0093
22	359.9	355.3	22.7	615.6	0.0016
23	307.7	213.7	97.5	108.2	0.0092
24	487.7	234.9	178.2	111.4	0.0090
25	580.0	265.3	209.8	139.9	0.0071
26	335.5	329.7	21.1	348.3	0.0029

Lisäksi mallinnusta varten on ladattu Geologian tutkimuskeskuksen Hakku-palvelusta [5] Maaperä 1:200 000 (maalajit)-aineisto [27], jonka avulla on rajattu jokilaaksojen alueet. Maaperä on vektoriaineisto, joka kuvaa yleistetysti Suomen maaperää. Aineiston avulla luodaan sallitut alueet, joiden sisällä joen migraatio on mahdollista simulaatiossa. Sallituiksi alueiksi on valittu jokia ympäröivät ”Karkearakeinen maalaji” -alueet (kuvat 12 ja 13), sillä nämä tarjoavat suotuisat olosuhteet joen meanderoinnille. Alueeseen kuuluu RT-luokituksen [1] mukaisesti maalajit, jotka on luokiteltu rakeiden läpimitan perusteella seuraavasti

- $> 1000\,\mathrm{mm}$: lohkareet
- $1000 - 60\,\mathrm{mm}$: kivet
- $60 - 2,0\,\mathrm{mm}$: sora
- $2,0 - 0,2\,\mathrm{mm}$: hiekka
- $0,2 - 0,06\,\mathrm{mm}$: karkea hieta.



Kuva 12: Oulankajoen sallittu alue on rajattu punaisella.



Kuva 13: Pulmankijoen sallittua alue on rajattuna punaisella.

Aineistoissa on käytetty EUREF-FIN-tasokoordinaattijärjestelmää ETRS-TM35FIN, joka on Suomen alueella yleisesti käytetty koordinaattijärjestelmä. Karttaprojektioiden avulla voidaan esittää kolmiulotteinen maapallon pinta kaksiulotteisella tasolla. ETRS-TM35FIN koordinaattijärjestelmä perustuu poikittaiseen Mercatorin projektioon, jossa maan vertausellipsoidin ympärille asetetaan lieriö, jonka pinnalle maapallon pinta projisoidaan. Kun lieriö avataan tasoksi, muodostuu suorakulmainen tasokoordinaatisto [16].

ETRS-TM35FIN on metrinen tasokoordinaatisto eli se on verrattavissa matematiikassa käytettävään karteesiseen koordinaatistoon. Koordinaatiston yksikkönä on siis metri ja sijainti kerrotaan muodossa (E, N). Itäkoordinaatti E kertoo sijainnin vaakasuunnassa ja pohjoiskoordinaatti N pystysuunnassa. Itäkoordinaatti saa keskimeridiaanilla eli noin 27 pituusasteen kohdalla arvon 500 000 metriä, ja kasvaa itään päin mentäessä. Pohjoiskoordinaatti taas on etäisyys päiväntasaajalta ja sen arvo kasvaa mentäessä pohjoiseen päin [16]. Tässä järjestelmässä esitettyjen koordinaattipisteiden väliset etäisyydet voidaan laskea suoraan euklidisellä etäisyyskaavalla. Näin ollen kahden koordinaattipisteen (E_1, N_1) ja (E_2, N_2) välinen etäisyys d on

$$d = \sqrt{(E_1 - E_2)^2 + (N_1 - N_2)^2}$$

ja tulos on metrejä.

4 Mallinnusmenetelmään liittyvää matemaattista teoriaa

Tässä luvussa tarkastellaan jokiuoman simulaatioon liittyviä matemaattisia käsitteitä. Luvussa käsitellyt asiat perustuvat lähteisiin [7, 11, 17, 23, 25, 26].

4.1 Pienimmän neliösumman menetelmä

Pienimmän neliösumman menetelmällä etsitään paras mahdollinen n-asteisen polynomin sovitus annettuihin datapisteiin. Menetelmä minimoi datapisteiden ja niihin sovitettavan polynomin välistä poikkeamaa.

Määritelmä 1. Olkoon $(x_1, y_1), (x_2, y_2) \ldots, (x_n, y_n)$ datapisteitä ja $\hat{y} = c_0 + c_1 x + c_2 x^2 + \cdots + c_n x^n$ n-asteinen polynominen malli. *Pienimmän neliösumman menetelmä* minimoi poikkeamien ϵ_i neliösummaa

$$\sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} [y_i - (c_0 + c_1 x_i + c_2 x_i^2 + \cdots + c_n x_i^n)]^2. \tag{1}$$

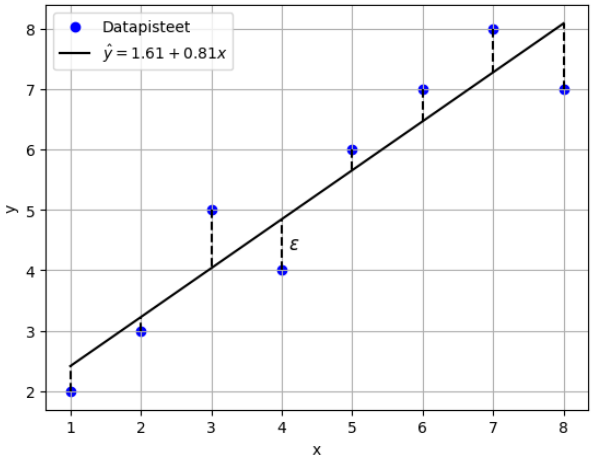
Tässä $c_0, c_1, c_2, \ldots, c_n$ ovat parametreja, jotka optimoidaan niin että sovitusvirhe ϵ_i olisi mahdollisimman pieni havaittujen arvojen y_i ja polynomimallin arvojen $\hat{y}_i$ välillä.

Esimerkki 1. Olkoon datapisteet $(1, 2), (2, 3), (3, 5), (4, 4), (5, 6), (6, 7), (7, 8), (8, 7)$. Sovitetaan näihin datapisteisiin lineaarinen malli $\hat{y} = c_0 + c_1 x$, jonka tuntemattomat parametrit c_0 ja c_1 määritetään minimoimalla poikkeamien ϵ_i neliösummaa

$$S = \sum_{i=1}^{8} \epsilon_i^2 = \sum_{i=1}^{8} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{8} [y_i - (c_0 + c_1 x_i)]^2$$

Parametrien c_0 ja c_1 arvot saadaan ratkaisemalla optimointitehtävä $\min S$

Annetuille datapisteille tehtävän $\min S$ ratkaisuksi saadaan $c_0 = 1.61$ ja $c_1 = 0.81$, jolloin sovitettu lineaarinen malli on muotoa $\hat{y} = 1.61 + 0.81x$.



Kuva 14: Datapisteisiin sovitettu lineaarinen malli $\hat{y} = 1.61 + 0.81x$.

4.2 Konvoluutio

Konvoluutio on matemaattinen operaatio, jolla voidaan vähentää epäsäännöllisyyksiä datassa. Tätä käytetään esimerkiksi signaalinkäsittelyssä kohinan poistamiseen ja signaalin tasoittamiseen. Konvoluutio voidaan suorittaa sekä diskreeteille että jatkuville funktioille.

Määritelmä 2. Olkoon f ja g diskreettejä funktioita, jotka on määritelty pisteissä $n \in \mathbb{Z}$. *Kahden diskreetin funktion f ja g välinen konvoluutio* määritellään konvoluutiosummana

$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m].$$

Määritelmä 3. Olkoon f ja g jatkuvia funktiota, jotka on määritelty pisteissä $t \in \mathbb{R}$. *Kahden jatkuvan funktion f ja g välinen konvoluutio* määritellään integraalina

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau.$$

Konvoluutioissa siis toinen funktio käännetään ja liikutetaan toisen funktion yli siten, että jokaisessa kohdassa lasketaan näiden funktioiden pisteittäinen tulo. Diskreetissä tapauksessa tulot summataan yhteen ja jatkuvassa tapauksessa otetaan integraali tuloista.

4.3 Savitzky-Golay-suodatin

Mallinnusmenetelmässä joen keskiviiva jaetaan tasamittaisiin osaväleihin. Jakoväli voidaan valita uomasta riippuen niin, että joen muodossa säilytetään riittävä tarkkuus, mutta samalla varmistetaan myös hyvä laskentateho.

Pistejonoon saattaa syntyä pieniä ei-toivottuja poikkeamia, jotka saattavat johtua esimerkiksi digitoinnista, keskiviivan muodostuksesta tai viivan pisteisiin jaosta. Algoritmia varten pistejono tasoitetaan, jotta pienet notkahdukset eivät aiheuta häiriötä laskelmiin ja tuloksiin.

Pisteiden tasoittamiseen käytetään Savitzky-Golay-suodatinta, joka perustuu konvoluutioon liikkuvalla välillä. Kullekin välille sovitetaan polynomi, jonka avulla suodatin tasoittaa dataa. Sovitus tehdään pienimmän neliösumman menetelmällä. Tämä prosessi suoritetaan jokaiselle datapisteelle, jolloin suodatin tasoittaa dataa konvoluutioperiaatteella. Näin Savitzky-Golay-suodattimella voidaan vähentää kohinaa datapisteissä, mutta säilyttää kuitenkin samalla tarkasti joen muoto kaarteineen.

Määritelmä 4. Olkoon $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ datapisteitä. *Savitzky-Golay -suodatin* sovittaa datapisteille polynomin liikkuvan välin yli, ja tämän sovituksen avulla datapisteet tasoitetaan konvoluutiolla

$$\hat{y}_i = \frac{\sum_{j=-k}^{k} C_j \times y_{i+j}}{n}, \tag{2}$$

missä n on normalisointitekijä ja tarkasteltavan pistevälin pituus on $2k + 1$. Kertoimet C_j ovat konvoluution painokertoimia, jotka määritetään polynomisovituksen perusteella.

Savitzky-Golay-suodattimen käyttö paransi huomattavasti meanderikaarteiden määrittelyä ja siten myös muita algoritmin osatehtäviä. Suodatin tasoittaa kaarteen pisteistä turhat poikkeamat, mutta silti samalla säilyttää kaarteen keskeisen muodon, kuten kuvan (15) tapauksessa nähdään.

Kuva 15: Savitzky-Golay-suodattimen avulla voidaan tasoittaa datapisteitä. Keskimmäinen piste saa uudeksi arvokseen sovitetun polynomin arvon keskikohdalla ja näin pistejonon tasoittuu yksinkertaisempaan muotoon. Kuvan tapauksessa käytetään kolmannen asteen polynomia ja $k = 3$.

4.4 Konveksisuus

Mallinnusmenetelmässä liikutetaan meanderikaarteiden pisteitä. Tätä varten jokiuomasta erotellaan kaarteet, eli kohdat, joissa jokiuoman kuperuussuunta muuttuu konveksista konkaaviin, ja päinvastoin.

Määritelmä 5. Joukko S on *konveksi joukko*, jos kahden mielivaltaisen joukon S pisteen x ja y yhdysjana sisältyy kokonaan joukkoon S, eli

$$\lambda x + (1-\lambda)y \in S, \quad \text{kaikilla } x, y \in S \text{ ja } \lambda \in (0,1).$$

Määritelmä 6. Olkoon $S \subset \mathbb{R}^n$ konveksi. Funktio $f : S \to \mathbb{R}$ on *konveksi funktio*, jos

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) - (1-\lambda)f(y) \quad \text{kaikilla } x, y \in S \text{ ja } \lambda \in (0,1).$$

Funktio f on *aidosti konveksi funktio*, jos sille pätee

$$f(\lambda x + (1-\lambda)y) < \lambda f(x) - (1-\lambda)f(y) \quad \text{kaikilla } x, y \in S, x \neq y, \text{ ja } \lambda \in (0,1).$$

Vastaavasti funktio f on *(aidosti) konkaavi*, jos $-f$ on (aidosti) konveksi.

Funktio on siis konveksi, jos minkä tahansa kahden kuvaajalta valitun pisteen välinen jana kulkee aina kuvaajan yläpuolella. Vastaavasti funktio on konkaavi, jos jana pysyy kuvaajan alapuolella (kuva 16).

Kuva 16: Vasemman puolen funktio on konveksi, koska minkä tahansa kahden pisteen välille piirretty suora pysyy kokonaan funktion alapuolella. Oikean puolen funktio taas ei ole konveksi koko matkalta, koska on mahdollista löytää pisteitä, joiden välille piirretty suora ei pysy funktion yläpuolella.

4.5 Käännepisteet

Käännepisteet ovat funktion pisteitä, joiden kohdalla kuvaajan kuperuussuunta muuttuu, eli kuvaaja muuttuu konveksista konkaaviin tai toisinpäin (kuva 17).

Määritelmä 7. Olkoon $f : (a, b) \to \mathbb{R}$ kaksi kertaa derivoituva. Piste $x \in (a, b)$ on funktion f *käännepiste*, jos $f''(x) = 0$ ja jos jollain $r > 0$ funktion $f''(x)$ arvot ovat vastakkaismerkkiset väleillä $(x - r, x)$ ja $(x, x + r)$.

Kuva 17: Käännepisteet ovat pisteitä, joiden kohdalla käyrän kuperuussuunta muuttuu. Näiden avulla voidaan määritellä meanderikaarteet jokiuoman keskiviivasta.

Koska algoritmissa pistejonona esitettävä reitti ei ole derivoituva, selvitetään käännekohdat kolmen peräkkäisen pisteen x_{i-1}, x_i ja x_{i+1} välisten vektorien

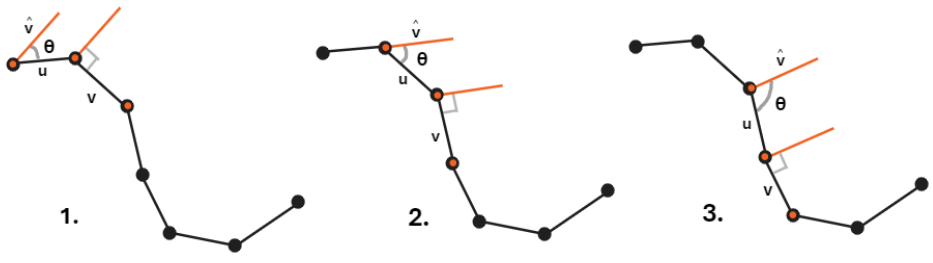
$$u_i = (x_i - x_{i-1}) \quad \text{ja} \quad v_i = (x_{i+1} - x_i),$$

sekä jälkimmäisen vektorin v_i normaalivektorin $\hat{v_i}$ avulla.

Käännepisteet x_i voidaan määrittää laskemalla, milloin pistetulo

$$u_i \cdot \hat{v_i} \tag{3}$$

vaihtaa etumerkkiä. Silloin kun pistetulo on positiivinen niin vektoreiden välinen kulma $\theta < 90°$, ja kun pistetulo on negatiivinen niin $\theta > 90°$ (kuva 18).



Kuva 18: Kuvassa nähdään oranssilla eri vaiheiden pisteet x_{i-1}, x_i ja x_{i+1} ja niiden väliset vektorit u_i ja v_i. Pistetulon $u_i \cdot \hat{v_i}$ avulla löydetään reitiltä kohdat, joissa kuperuussuunta muuttuu. Kahdessa ensimmäisessä vaiheessa kulma θ on alle 90 astetta ja käännekohta löytyy kohdassa kolme, kun kulma muuttuu yli 90 asteiseksi.

Algoritmissa meanderikaarteet määritellään niistä pisteistä, jotka jäävät käännepisteiden väliin.

4.6 Kaarevuus

Kaarevuus κ kuvaa käyrän taipumista tietyssä pisteessä. Se kertoo, kuinka nopeasti käyrä muuttaa suuntaansa kyseisessä pisteessä.

Määritelmä 8. Olkoon x_1 ja x_2 sileän käyrän $\gamma \in \mathbb{R}^2$ kaksi pistettä. Kaaren pituutta pisteiden x_1 ja x_2 välillä merkitään Δs ja pisteiden tangenttivektorien välistä kulmaa merkitään $\Delta\theta$. *Käyrän γ kaarevuus pisteessä* x_1 saadaan raja-arvosta

$$\kappa(x_1) = \lim_{x_2 \to x_1} \frac{\Delta\theta}{\Delta s} = \lim_{\Delta s \to 0} \frac{\Delta\theta}{\Delta s}.$$

Jos γ on r-säteinen ympyrä, niin tällöin

$$\Delta s = r\Delta\theta.$$

Ympyrän kaarevuudeksi κ saadaan tällöin kaarevuuden määritelmän mukaan

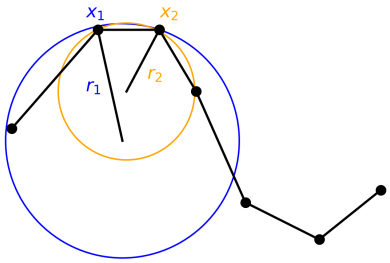
$$\lim_{\Delta s \to 0} \frac{\Delta\theta}{\Delta s} = \lim_{\Delta s \to 0} \frac{\Delta\theta}{r\Delta\theta} = \frac{1}{r}.$$

Näin ollen ympyrän kehän kaarevuus on kaikissa pisteissä vakio $\frac{1}{r}$.

Myös käyrän kaarevuus voidaan ajatella ympyrän säteen käänteislukuna, jos ympyrä sovitetaan käyrälle tarkasteltavaan kohtaan. Koska mallinnusmenetelmässä joen reitti esitetään pistejonona, niin pisteiden x_i kaarevuutta tarkastellaan kolmen peräkkäisen pisteen x_{i-1}, x_i ja x_{i+1} kautta kulkevan ympyrän avulla

$$\kappa(x_i) = \frac{1}{r_i}. \tag{4}$$

Kun kaarevuus ilmaistaan ympyrän säteen käänteislukuna, niin tällöin mitä suurempi ympyrän säde on, sitä pienempi on kaarevuus, ja toisinpäin (kuva 19).



Kuva 19: Kuvassa on esitettynä pisteille x_1 ja x_2 sovitetut ympyrät. Nähdään että pienempi säde tarkoittaa suurempaan kaarevuutta ja vastaavasti suurempi säde pienempää kaarevuutta.

Kaarevuudet normalisoidaan mallinnusmenetelmässä meanderikaarteittain, mikä tarkoittaa, että jokaiselle pisteelle lasketaan kaarevuusarvo suhteessa kyseisen kaarteen j suurimpaan kaarevuusarvoon $\frac{\kappa(x_i)}{\max \kappa_j}$. Normalisointi suhteuttaa kaarevuusarvot niin, että niitä voidaan vertailla keskenään riippumatta siitä, kuinka suuresta tai pienestä kaarteesta on kyse.

5 Migraation mallinnus

Tässä luvussa käsitellään meanderoivan joen migraation matemaattista mallintamista. Aluksi esitellään alkuperäinen ChaRMigS-menetelmä, jonka jälkeen tarkastellaan tulevaisuuden migraation mallintamiseen kehitettyä menetelmää.

5.1 ChaRMigS-menetelmä

ChaRMigS (*The Channel Reverse Migration Simulation*) on Parquer ym. [21] esittämä joen migraation simulaatio, joka etenee ajassa taaksepäin mallintaen joen mahdollisia reittejä historiassa. Jokaisen simulaation lopputulos vaihtelee hieman, sillä menetelmä sisältää satunnaisuutta kaarteiden migraation osalta. Algoritmi pyrkii myös integroimaan irronneita juoluoita takaisin osaksi uomaa, jos joen pääuoma päätyy tarpeeksi lähelle niitä.

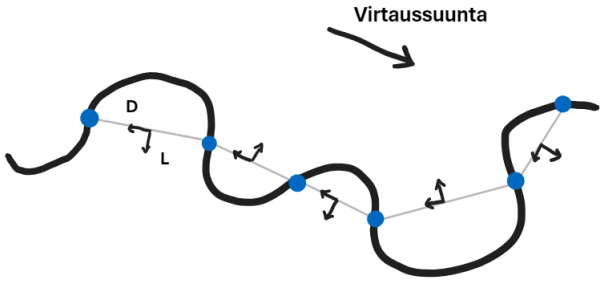
Parquer ym. [21] tutkivat Yhdysvalloissa sijaitsevaa Missisippi jokea, jonka historiallisista reiteistä oli käytettävissä dataa vuodesta 1765 lähtien. He tarkastelivat yhteyttä joen geometrian, erityisesti jokikaarteiden kaarevuusarvojen vaikutusta virtaus- ja sivuttaissuuntaiseen migraatioon. Kaarevuusarvojen ja migraatioiden välillä ei havaittu suoraa korrelaatiota, mutta kuitenkin kaarteittain normalisoitujen kaarevuusarvojen ja migraation välillä löydettiin yhteyksiä. Tämä siis tarkoittaa, että jokiuoman kaarevimmat kohdat eivät liikkuneet nopeammin muihin kaarteisiin verrattuna, mutta yleisesti kaarteiden huippukohdat liikkuivat eniten.

Tulokset osoittivat, että joissakin kaarteissa normalisoidun kaarevuuden kasvaessa migraatio oli suurempaa sivuttaissuuntaan ja samalla virransuuntainen liike väheni. Osassa kaarteissa taas ilmeni päinvastainen ilmiö, eli kaarevuuden kasvaessa migraatio lisääntyi virtaussuuntaan ja sivuttaissuuntainen liike väheni. Jotkin kaarteet eivät osoittaneet mitään tuloksia, mutta näissä ei myöskään ollut silmin nähtävissä liikettä. Lisäksi havaittiin tapauksia, joissa migraation kehitys oli negatiivista eli takautuvaa. ChaRMigS-menetelmä kehitettiin mallintamaan näitä ilmiöitä ja siten ennustamaan jokiuoman muutoksia historiassa.

Menetelmässä määritellään alkuun meanderikaarteet j. Tämän jälkeen lasketaan jokiuoman pisteille x_i kaarevuudet $\kappa(x_i)$ kolmen peräkkäisen pisteen x_{i-1}, x_i ja x_{i+1} kautta määritellyn ympyrän säteen käänteislukuna $\frac{1}{r_i}$. Lisäksi kaarteiden maksimikaarevuudet $\max \kappa_j$ määritellään kaarevuusarvojen normalisointia varten.

Jokaiselle kaarteelle j määritellään liikkumissuunnat virtaussuuntaan D_j (*downstream*) ja sivuttaissuuntaan L_j (*lateral*), joiden mukaisesti kaarteiden migraatio tapahtuu ajassa taaksepäin. ChaRMigS-menetelmässä suunta D_j on samansuuntainen käännepisteitä yhdistävän suoran kanssa ja osoittaa joen yläjuoksulle virtaussuun-

taa vasten. Sivuttainen suunta L_j on kohtisuorassa suuntaa D_j vasten ja osoittaa kaarteen sisään (kuva 20).



Kuva 20: Liikkumasuunnat määritellään virtaussuunnassa D käännepisteiden välisen suoran suuntaisesti ja sivuttaissuunnassa L kaarteen sisäosaa kohti. Näin jokiuoman liikettä voidaan mallintaa ajassa taaksepäin.

Simulaatiossa kaarteiden liike määräytyy migraatiomallin mukaan, jonka Parquer ym. [21] kehittivät meanderoivan jokiuoman migraatiota koskevien tutkimustulosten pohjalta. Kaarteille määritellään mallia varten maksimaaliset arvot sivuttaiselle ja alavirtaan tapahtuvalle muutokselle o_D ja o_L. Koska todellisessa ympäristössä sivuttais- ja virtaussuuntaan tapahtuvan migraation määrä vaihtelee kaarteiden välillä, niin maksimiarvot kaarteiden siirtymälle määrätään todennäköisyysjakaumasta. Todennäköisyysjakaumalla saadaan lisättyä vaihtelevuutta kaarteiden migraatioihin ja tällöin malli kuvaa paremmin reaalimaailman tilannetta. Tähän on käytetty Gaussin jakaumaa $N(\mu, \sigma^2)$, johon voidaan asettaa arvot keskimääräiselle liikkumalle μ ja liikkuman varianssille σ^2.

Maksimiliikkumien määrittelyn avulla simulaation toimintaa voidaan ohjata siten, että kaarteiden migraatio tapahtuu samanaikaisesti molempiin suuntiin, jolloin

$$o_D \sim N(\mu, \sigma^2) \quad \text{ja} \quad o_L \sim N(\mu, \sigma^2)$$

tai ainoastaan virtaussuuntaan, jolloin

$$o_D \sim N(\mu, \sigma^2) \quad \text{ja} \quad o_L = (0, \ldots, 0)$$

tai ainoastaan sivuttaissuuntaan, jolloin

$$o_D = (0, \ldots, 0) \quad \text{ja} \quad o_L \sim N(\mu, \sigma^2).$$

Seuraavaksi määritellään tasoitusparametrit s_D ja s_L, jotka valitaan tasajakaumista $U(0, 2o_D)$ ja $U(0, 2o_L)$, eli ne saavat satunnaiset arvot nollan ja kaksinkertaisen maksimiliikkuman väliltä. Tasoitusparametrit vaikuttavat liikkuma-arvojen suuruuteen ja mahdollistavat myös pisteiden negatiivisen siirtymän. Tämä vaikuttaa ajanjaksojen välillä migraation nopeuteen.

Lisäksi vaihtelevuutta meanderikaarteiden liikkumiseen lisätään positiivisen tai negatiivisen suuntapainotuksen w_j avulla, joka otetaan sattumanvaraisesti diskreetistä joukosta $\{-1, 1\}$. Tämä parametri vaikuttaa siihen, onko pisteiden liikkuminen kaarteessa suurempaa virtaus- vai sivuttaissuuntaan riippuen kaarevuuden arvosta.

Parametrien asetusten jälkeen liikkuma-arvot $O_D(x_i)$ ja $O_L(x_i)$ lasketaan jokiuoman kaarteiden pisteille Parquer ym. [21] kehittämien migraatioyhtälöiden

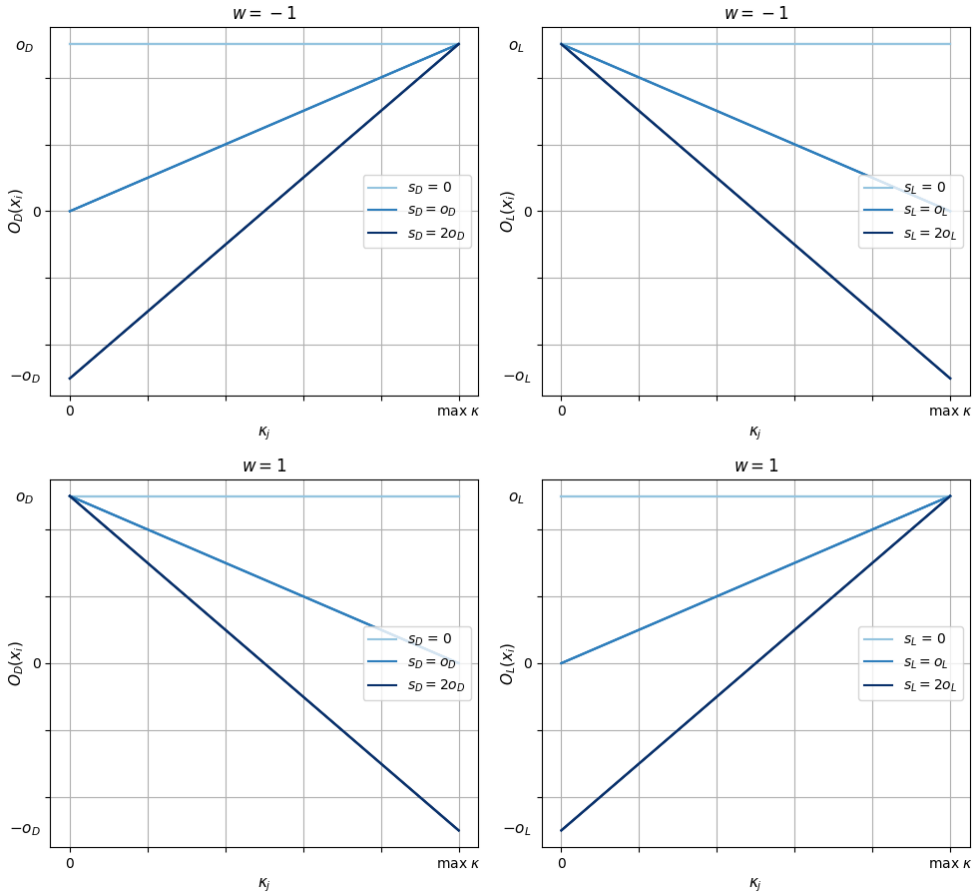
$$O_D(x_i) = \begin{cases} o_{D_j} - s_{D_j} \times \frac{\kappa(x_i)}{\max \kappa_j}, & \text{kun } w_j > 0 \\ o_{D_j} - s_{D_j} \times \left(1 - \frac{\kappa(x_i)}{\max \kappa_j}\right), & \text{kun } w_j < 0 \end{cases} \tag{5}$$

ja

$$O_L(x_i) = \begin{cases} o_{L_j} - s_{L_j} \times \left(1 - \frac{\kappa(x_i)}{\max \kappa_j}\right), & \text{kun } w_j > 0 \\ o_{L_j} - s_{L_j} \times \frac{\kappa(x_i)}{\max \kappa_j}, & \text{kun } w_j < 0 \end{cases} \tag{6}$$

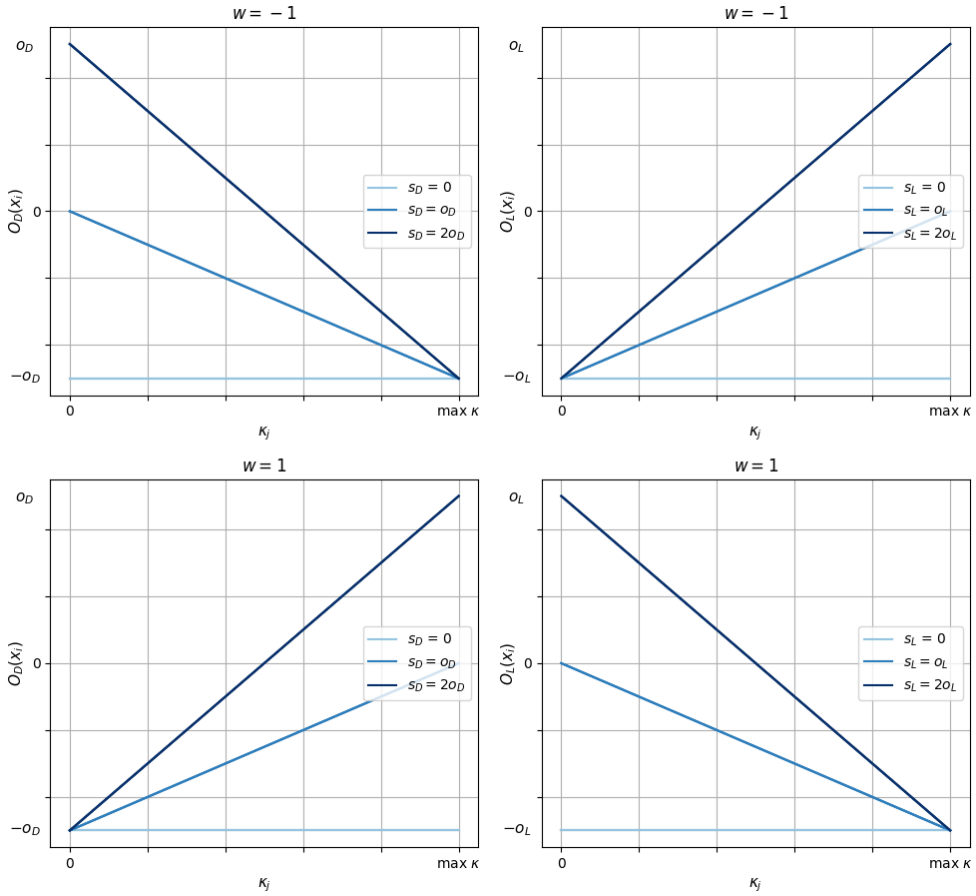
mukaan.

Jos w_j on positiivinen, niin virtaussuunnassa tapahtuva migraatio O_D vähenee ja sivuttanen migraatio O_L kasvaa, kun kaarevuus kasvaa. Vastaavasti, jos w_j on negatiivinen niin O_D kasvaa ja O_L vähenee, kun kaarevuus kasvaa. Lisäksi, jos s_D tai s_L on huomattavasti suurempi kuin o_D tai o_L ja kaarevuuden vaikutus on vähäistä, niin liike pisteelle voi olla myös negatiivista (kuva 21).



Kuva 21: Kuvassa nähdään kaarevuuden κ_j vaikutus liikkuma-arvoihin eri w:n arvoilla. Vasemmanpuoleiset kuvaajat esittävät yhtälön (5) arvoja D-suuntaan tapahtuvalle liikkeelle ja oikeanpuoleiset yhtälön (6) arvoja L-suuntaan tapahtuvalle liikkeelle. Kun tasoitusparametrit s_D ja s_L saavat suurempia arvoja kuin mitä o_D tai o_L ovat, niin liike voi saada myös negatiivisen arvon, jolloin liikutettava piste siirtyy taaksepäin. Pienillä s_D ja s_L arvoilla liike on suurinta.

ChaRMigS-menetelmällä voidaan mallintaa myös negatiiviseen suuntaan tapahtuvaa migraatiota, mikä tehdään valitsemalla o_D ja o_L normaalijakaumasta $N(0, \sigma^2)$. Tällöin maksimiliikkuma voi olla yhtä todennäköisesti joko positiivista tai negatiivista, mikä lisää takautuvan migraation esiintymistä verrattuna edelliseen määrittelyyn. Negatiivisen maksimiliikkuman tapauksessa kaarevuuden kasvaessa liike on suurempaa negatiiviseen suuntaan (kuva 22).



Kuva 22: Kuvassa nähdään kaarevuuden κ_j vaikutus liikkuma-arvoihin eri w:n arvoilla, jos maksimiliikkumana o_D tai o_L on negatiivinen luku. Vasemmanpuoleiset kuvaajat esittävät yhtälön (5) arvoja D-suuntaan tapahtuvalle liikkeelle ja oikeanpuoleiset yhtälön (6) arvoja L-suuntaan tapahtuvalle liikkeelle. Kun tasoitusparametrit s_D ja s_L saavat suurempia arvoja kuin mitä o_D tai o_L ovat, niin liike voi saada myös positiivisen arvon.

Jokiuoman liikuttamisen lisäksi uomaan pyritään kiinnittämään siitä irronneita juoluoita. Juoluoiden liittäminen tehdään perustuen niille arvioituun irtaantumisikään sekä juoluan etäisyyteen pääuomasta. Koska irtaantumisikää on vaikea tietää tarkasti, juoluan ikä määritellään satunnaisesti niille arvioidun minimi- ja maksimiiän väliltä. Kun algoritmissa saavutetaan irtoamisikä, yritetään juolua integroida osaksi uomaa. Juoluan päiden etäisyys lasketaan pääuomaan ja jos laskettu etäisyys on sallituissa rajoissa niin juolua liitetään jokeen.

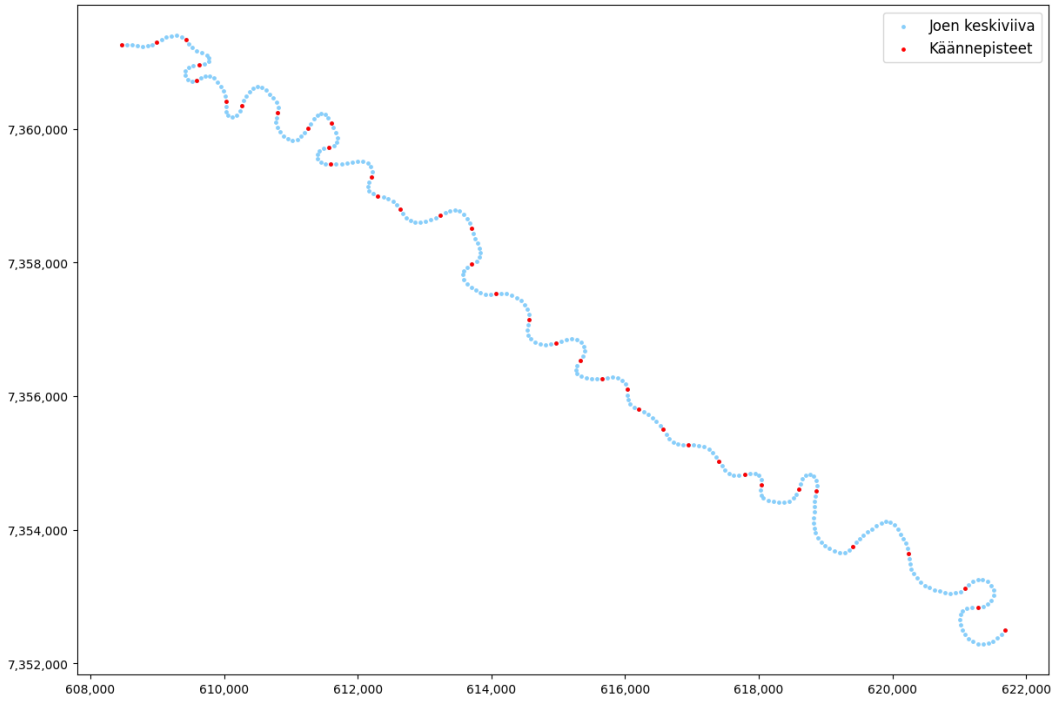
Mallinnus keskittyy siis puhtaasti jokiuoman geometriaan, eikä ota huomioon esimerkiksi maaston muotoja tai hydrologisia tekijöitä. Jokiuoman kaarteiden kaarevuusarvoihin perustuen malli pyrkii ennustamaan uoman kehityspolkuja historiassa. Lisäksi stokastisuus tuo luonnollista vaihtelevuutta simulaatioihin.

5.2 Jokiuoman simulaatio ajassa eteenpäin

Tutkielman tavoitteena on soveltaa yllä esitettyä ChaRMigS-menetelmää jokiuoman migraation mallintamiseen ajassa eteenpäin. Tässä mallinnusmenetelmässä liikkumissuunnat käännetään tulevaisuuden suuntaan ja lisätään ehto uusien juoluoiden muodostumiselle. Lisäksi joen migraatiolle asetetaan sallittu alue, jonka sisällä joki saa liikkua ja näin pyritään saamaan realistisempia simulaatiotuloksia.

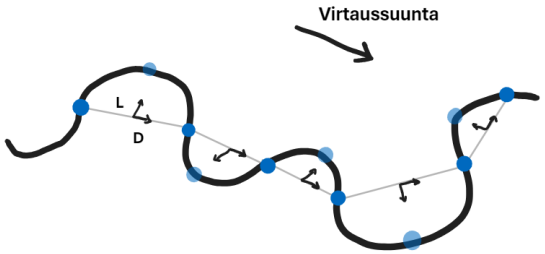
Joen tulevaisuutta mallintavalle algoritmille annetaan syötteeksi haluttujen simulaatioiden lukumäärä N_s, jokiuoman keskiviivan koordinaattipisteet, kaarteiden keskimääräinen liikkuma-arvo μ ja liikkuma-arvon varianssi σ^2, etäisyys uuden juoluan muodostumiselle sekä maankäyttöaineistosta muodostettu sallittu alue S.

Kuten ChaRMigS-menetelmässä, algoritmi erottelee alkuun keskiviivasta meanderikaarteet j käännepisteiden avulla. Käännepisteet, eli kuperuussuunnan muutoskohdat, etsitään kolmen peräkkäisten pisteiden välisten vektorien pistetulolla, kuten kuvassa 3. Meanderikaarteet määritellään näiden käännepisteiden avulla niin, että käännepisteiden välissä olevat pisteet muodostavat yhden kaarteen (kuva 23).



Kuva 23: Oulankajoen keskiviivan pistejako nähdään kuvassa sinisellä ja algoritmin löytämät käännepisteet on merkitty kuvaan punaisella. Käännepisteiden väliin jäävät pisteet muodostavat meanderikaarteet. Menetelmä onnistuu myös erottamaan tuplakaarteet, joissa kaarre on alkanut jakautua kahteen suuntaan.

Tämän jälkeen kaarteille j määritellään migraatiosuunnat D_j ja L_j (kuva 24). Virtaussuuntainen liike D_j tapahtuu nyt kohti alavirtaa, koska joen muutosta halutaan mallintaa tulevaisuuteenpäin. Suunta D_j määritellään samoin kuten ChaRMigS-menetelmässä, eli samansuuntaisena peräkkäisten käännepisteiden välisen suoran kanssa. Sivuttaissuuntainen liike L_j määritellään hieman poiketen ChaRMigS-algoritmin määritelmästä, joka käsittelee suuntaa L_j kohtisuorassa suuntaan D_j nähden. Sivuttaissuuntainen liike L_j määritellään nyt käännepisteiden välisen suoran keskipisteestä kohti meanderikaarteen huippupistettä. Kaarteen huippupisteeksi on määritelty kaarteen piste, jonka etäisyys on suurin käännepisteitä yhdistävästä suorasta. Näin saadaan kuvattua paremmin sivuttaista liikettä esimerkiksi kääntyneiden kaarteiden tapauksessa.

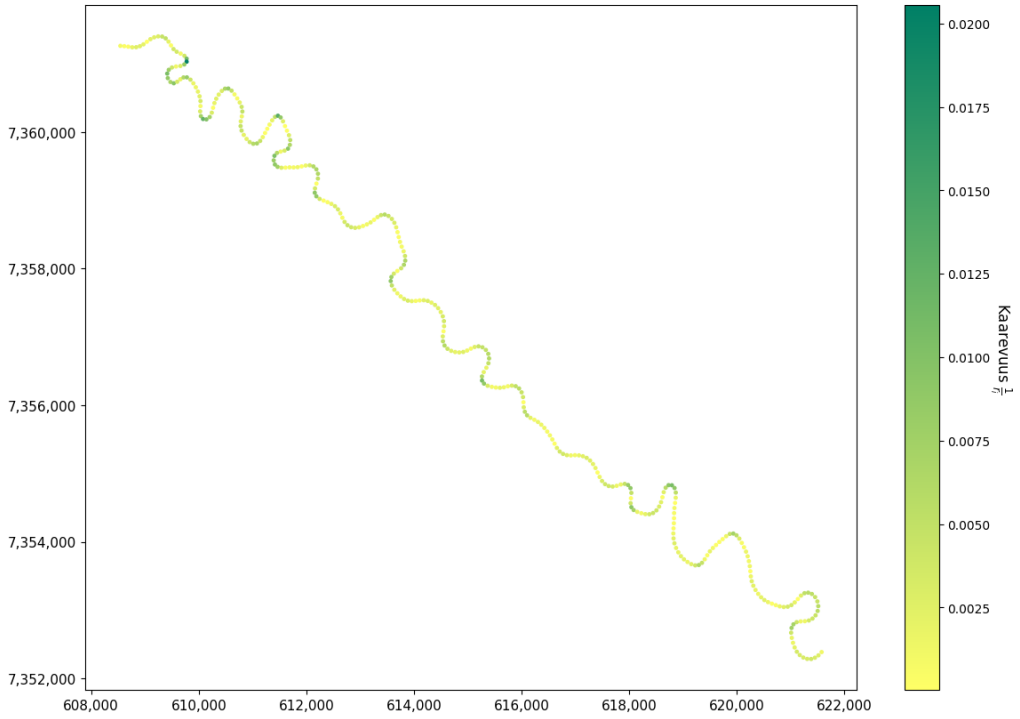


Kuva 24: Liikkumasuunnat määritellään virtaussuunnassa D samansuuntaisesti käännepisteiden välisen suoran kanssa ja sivuttaissuunnassa L kaarteen huippupistettä kohti.

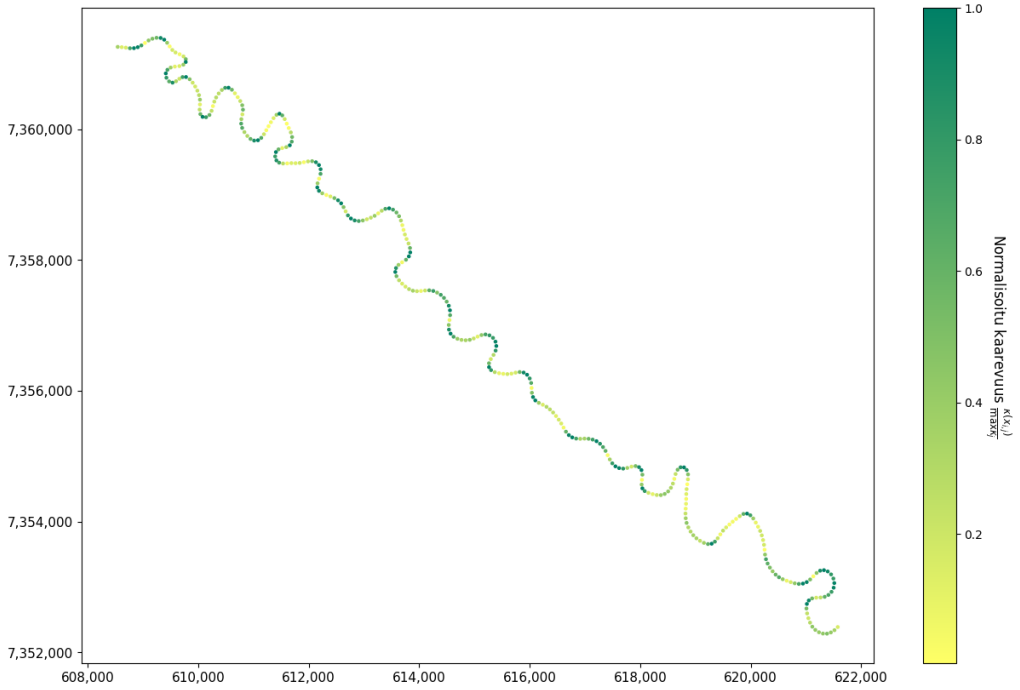
Samaan tapaan kuin ChaRMigS-menetelmässä, jokaiselle kaartelle j määrätään:

- Maksimi liikkuma-arvot o_{D_j} ja o_{L_j} normaalijakaumasta $N(\mu, \sigma^2)$, jossa μ on arvo keskimääräiselle liikkumalle ja σ liikkuman varianssi.
- Tasoitusparametrit s_{D_j} ja s_{L_j} tasajakaumista $U(0, 2o_{D_j})$ ja $U(0, 2o_{L_j})$.
- Suunnanpainotusarvot w_j diskreetistä joukosta $\{-1, 1\}$.

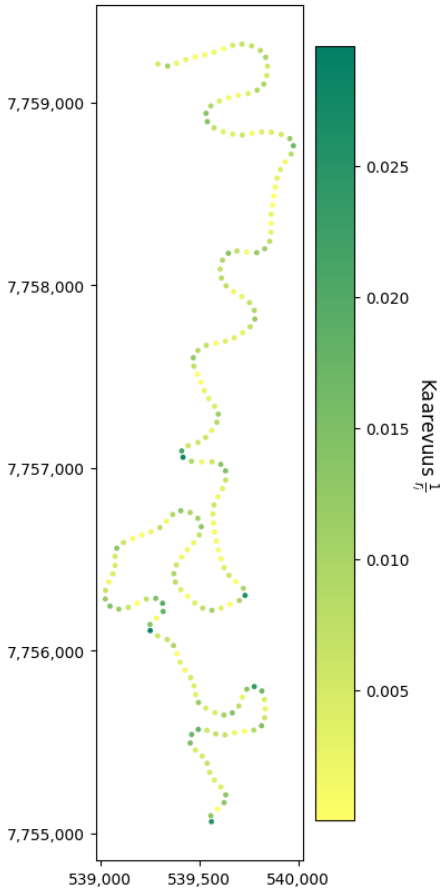
Seuraavaksi pisteille x_i lasketaan kaarevuusarvot kolmeen peräkkäiseen pisteeseen sovitetun ympyrän käänteislukuna $\kappa(x_i) = \frac{1}{r_i}$. Kuvissa 25 ja 27 nähdään kaarevuusarvot Oulankajoen ja Pulmankijoen keskiviivojen pisteille. Lisäksi menetelmässä määritellään kaarteiden maksimikaarevuusarvot $\max \kappa_j$, joiden avulla kaarevuudet voidaan normalisoida kaarteittain $\frac{\kappa(x_i)}{\max \kappa_j}$. Kuvissa 26 ja 28 nähdään normalisoidut kaarevuusarvot.



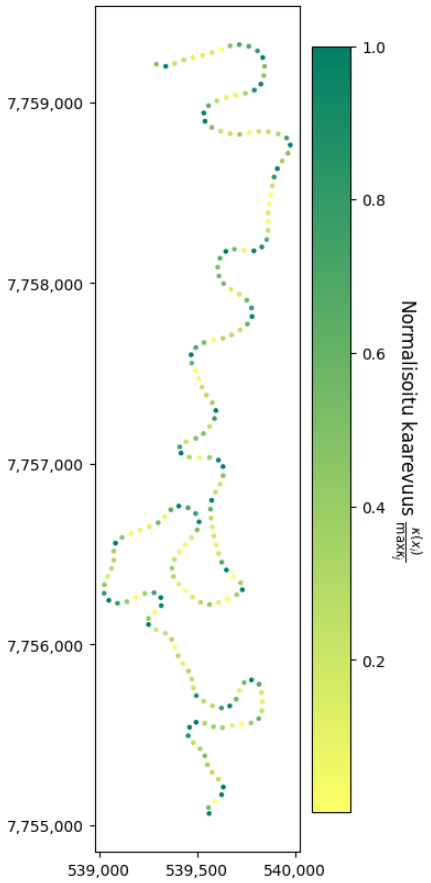
Kuva 25: Oulankajoen kaarevuusarvot laskettuna kolmeen peräkkäiseen pisteeseen sovitetun ympyrän käänteislukuna $\frac{1}{r_i}$.



Kuva 26: Oulankajoen kaarevuusarvot normalisoituna kaarteiden maksimikaarevuuden suhteen. Normalisointi korostaa kaarevuusarvoja ja kaarteiden huippuja riippumatta niiden alkuperäisestä suuruudesta.



Kuva 27: Pulmankijoen kaarevuusarvot laskettuna kolmeen peräkkäiseen pisteeseen sovitetun ympyrän käänteislukuna $\frac{1}{r_i}$.



Kuva 28: Kaarevuusarvot normalisoituna kaarteiden maksimikaarevuuden suhteen.

Kun algoritmille tarvittavat parametrit on määritelty, kaarteiden j pisteitä x_i siirretään ChaRMigS-menetelmän migraatiomallin yhtälöiden (5) ja (6) mukaisesti. Jos w_j on positiivinen, niin virtaussuunnassa tapahtuva migraatio O_{D_j} vähenee ja sivuttanen migraatio O_{L_j} kasvaa, kun kaarevuus kasvaa. Näin voidaan kuvata tilannetta, jossa joen kaarre levenee, kun suurin eroosio kohdistuu kaarteen huippukohtaa kohden. Vastaavasti, jos w_j on negatiivinen niin O_{D_j} kasvaa ja O_{L_j} vähenee, kun kaarevuus kasvaa. Tämä kuvaa tilannetta, jossa kaarteen huippukohta on vastustuskykyinen eroosiolle ja vesi ohjautuu nopeammin kaarteen läpi, jolloin virran suuntainen eroosio on suurempaa. Jokiuoman pisteen liikkumista havainnollistetaan esimerkissä 2.

Esimerkki 2. Tarkastellaan jokiuoman pisteen $x = (610\,300, 7\,360\,500)$ liikkumista TM35FIN-koordinaatiston mukaan. Koska TM35FIN on tasokoordinaatisto, jonka

yksikkönä on metri niin voidaan sen ajatella toimivan xy-koordinaatiston tavoin.

Oletetaan, että suunta D on virran mukainen käännepisteiden $(610\,200, 7\,360\,400)$ ja $(610\,600, 7\,360\,200)$ välillä. Näin ollen

$$D = (E_1 - E_2, N_2 - N_1) = (610\,600 - 610\,200, 7\,360\,200 - 7\,360\,400) = (400, -200).$$

Suunnan D pituus on

$$||D|| = \sqrt{400^2 + (-200)^2} = 200\sqrt{5},$$

jonka avulla saadaan normalisoitu suunta

$$D = \Big(\frac{400}{200\sqrt{5}}, \frac{-200}{200\sqrt{5}}\Big) \approx (0.894, -0.447).$$

Suunta L on käännepisteiden välisen suoran keskipisteen ja huippupisteen välinen suunta. Olkoon huippupiste $(610\,500, 7\,360\,600)$ ja käännepisteiden välisen suoran keskipiste $(610\,400, 7\,360\,300)$. Suunta L on siis

$$L = (100, 300)$$

ja sen pituus on

$$||L|| = \sqrt{100^2 + 300^2} = 100\sqrt{10}.$$

Normalisoiduksi suunnaksi L saadaan

$$L = \Big(\frac{100}{100\sqrt{10}}, \frac{300}{100\sqrt{10}}\Big) \approx (0.316, 0.949).$$

Oletetaan, että todennäköisyysjakaumista saadut arvot ovat $o_D = 25$, $o_L = 18$, $s_D = 3$, $s_L = 42$ ja $w = -1$. Oletetaan että pisteen $x = (610\,300, 7\,360\,500)$ normalisoitu kaarevuus tälle pisteelle on $0,75$. Tällöin mallin (5) ja (6) mukaan pisteelle x saadaan liikkuma-arvot

$$O_D(x) = 18 - 3 \times (1 - 0.75) = 17.25$$

ja

$$O_L(x) = 25 - 42 \times 0.75 = -6.5.$$

Näin ollen koordinaattipistettä $x = (610\,300, 7\,360\,500)$ liikutetaan 17.25 metriä suuntaan D ja 6.5 metriä taaksepäin suuntaan L, sillä $O_L(x) < 0$. Uuden pisteen $x' = (E', N')$ koordinaateiksi saadaan

$$\begin{aligned} E' &= E + (O_D(x) \times D_0 + O_L(x) \times L_0) \\ &= 610\,300 + (17.25 \times 0.894 - 6.5 \times 0.316) \\ &\approx 610\,313.37 \end{aligned}$$

ja

$$\begin{aligned} N' &= N + (O_D(x) \times D_1 + O_L(x) \times L_1) \\ &= 7\,360\,500 + (17.25 \times -0.447 - 6.5 \times 0.949) \\ &\approx 7\,360\,486.12. \end{aligned}$$

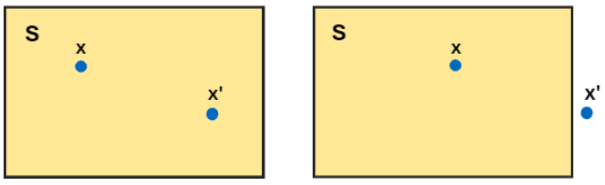
Näin ollen uusi koordinaattipiste on $x' = (610\,313.37, 7\,360\,486.12)$.

Mallinnusmenetelmään on lisätty sallittu alue, joka määrää alueen, jonka sisäpuolella jokiuoma saa liikkua simulaation aikana. Tutkielmassa sallitut alueet on muodostettu Maaperä -aineistoista rajaamalla ”karkea-aines” -alueet kuvaamaan jokilaaksojen alueita (kuvat 12 ja 13). Tämä rajaus mahdollistaa realistisemman joen simulaation, sillä se määrittää alueen, jonka sisällä jokiuomaa pystyy muokkautumaan maaperän ominaisuuksien mukaisesti. Sallittu alue S voidaan määritellä seuraavasti:

$$S = \{(E, N) \,|\, (E, N) \text{ kuuluu rajattuun ”karkea-aines” -alueeseen}\}.$$

Algoritmi tarkistaa ennen uuden pisteen asettamista, että kuuluuko liikutettava piste x_i' joukkoon S (kuva 29). Jos ehto toteutuu niin pistettä voidaan siirtää, ja jos ehto ei toteudu niin pistettä ei liikuteta kyseisellä iteraatiokierroksella:

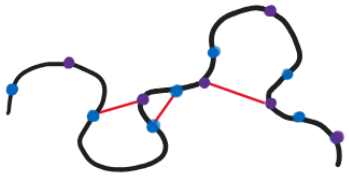
$$\text{uusi } x_i = \begin{cases} x_i' & \text{jos } x_i' \in S \\ x_i & \text{jos } x_i' \notin S. \end{cases}$$



Kuva 29: Piste x on alkuperäinen piste ja piste x' on liikutettu piste. Vasemmalla piste x' pysyy sallitulla alueella S, joten *uusi* $x = x'$. Oikealla taas piste x' ei pysy sallitulla alueella, joten *uusi* $x = x$, eli piste pysyy paikallaan.

Lisäksi mallinnusmenetelmässä on uusien juoluoiden muodostumiselle ehto, mikä mahdollistaa joen kehityksen ennustamisen tarkemmin. Juoluan muodostumiseksi algoritmi vertailee käännepisteiden ja huippupisteiden välisiä etäisyyksiä ja katsoo ovatko ne tarpeeksi lähellä toisiaan, jotta juolua voisi muodostua. Jos jokin

etäisyyksistä on alle määritellyn arvon, uusi juolua muodostuu. Tällöin ehdon täyttävien pisteiden väliin jäävät pisteet jätetään simulaation ulkopuolelle seuraavalla kierroksella ja jokiuoma suoristuu.



Kuva 30: Algoritmissa uusi juolua voi syntyä käännepisteiden (sininen) tai huippupisteiden (violetti) väliin, jos jokin etäisyyksistä käännepisteiden välillä, huippupisteiden välillä tai käänne- ja huppupisteiden välillä on alle annetun arvon. Tällöin jokiuoma suoristuu pisteiden välillä ja ulkopuolelle jäävät pisteet muodostavat juoluan. Mahdolliset rajan alittavat kohdat on havainnollistettu kuvaan punaisella.

Algoritmin lopuksi liikutetut pisteet kootaan takaisin yhtenäiseksi viivaksi, joka jaetaan uudelleen tasamittaisiin osaväleihin seuraavaa iteraatiokierrosta varten. Näin joki pysyy muodossaan, eikä yksittäiset pisteet pääse liikkumaan luonnottomia määriä ja muuttamaan joen muotoa liikaa.

Jokaisella iteraatiokierroksella määritellään käännepisteet uudelle jokiuomalle ja määritellään kaarteet uudelleen. Myös algoritmin parametrit o_{D_j}, o_{L_j}, s_{D_j}, s_{L_j} ja w_j valitaan uudelleen, lasketaan kaarevuudet $\kappa(x_i)$ ja määritellään uudet liikkumissuunnat D_j ja L_j. Näin varmistetaan kaarteiden jatkuva kehitys sekä mahdollisuus uusien kaarteiden muodostumiselle.

Yhdistämällä nämä toiminnot saadaan luotua simulaatio, joka ennustaa jokiuoman tulevaa migraatiota perustuen uoman keskiviivan geometriaan. Menetelmä huomioi kaarteiden vaeltamisen, juoluoiden muodostumisen ja sallitun liikkumisalueen.

5.3 Algoritmi

Alla esitetään algoritmi tutkielmassa kehitetylle menetelmälle.

Input:
Simulaatioiden lukumäärä N_s ;
Joen keskiviiva pistejonona x_i ;
Keskimääräinen liikkumisarvo (m) μ ;
Liikkumisarvon varianssi (m) σ^2 ;
Etäisyys käännepisteiden välillä, jolloin muodostuu uusi juolua d (m);
Sallittu alue S;
Output: Simulaatio jokiuoman mahdolliselle migraatiolle
for $s \leftarrow 0$ **to** N_s **do**
    Määritellään käännepisteet ;
    Määritellään meanderikaarteet j ;
    Määritellään meanderikaarteiden lukumäärä N_j ;
    Määritellään huippupisteet ;
    Lasketaan kolmen peräkkäisten pisteiden määrittämien ympyröiden säteet r_i;
    Lasketaan pisteille kaarevuusarvot $\frac{1}{r_i}$;
    Määritellään migraatiosuunnat D_j ja L_j ;
    Määritellään maksimi liikkumismäärät o_{D_j} ja o_{L_j} normaalijakaumasta $N(\mu, \sigma^2)$ tai nollavektoriksi $(0, 0, \ldots, 0)$;
    Määritellään tasoitusparametrit s_{D_j} ja s_{L_j} tasajakaumasta U ;
    Määritellään painotus w_j diskreetistä joukosta $\{-1, 1\}$;
    for $j \leftarrow 0$ **to** N_j **do**
        Valitaan meanderikaarteen j pisteet x_i ;
        Määritellään liikkuma-arvot $O_D(x_i)$ ja $O_L(x_i)$;
        for $x_i \in j$ **do**
            Liikutetaan pistettä x_i kaarteelle j määriteltyjen suuntien ja liikkuma-arvojen mukaan;
            Tarkistetaan, kuuluuko liikutettu piste sallittuun alueeseen S ;
        end
    end
    Päivitetään uusi keskiviiva;
    Tarkistetaan, muodostuuko uusia juoluoita;
    if *uusi juolua* **then**
        Päivitetään uusi keskiviiva poistamalla juoluan muodostavat pisteet ;
    end
    Yhdistetään pistejono yhtenäiseksi käyräksi;
    Jaetaan käyrä uudelleen tasamittaisiin osaväleihin;
end

Algoritmin tuottamia simulaatiotuloksia nähdään luvussa 6.

Taulukko 4: Merkinnät

j	Meanderikaarre
x_i	Joen keskiviivan piste
$o_{D_j} \sim N(\mu, \sigma^2)$	D-suuntaan tapahtuva maksimi liike kaarteessa j
$o_{L_j} \sim N(\mu, \sigma^2)$	L-suuntaan tapahtuva maksimi liike kaarteessa j
$s_{D_j} \sim \mathrm{Tas}(0, 2o_{D_j})$	Tasoitusparametri D-suunnan liikkeelle kaarteessa j
$s_{L_j} \sim \mathrm{Tas}(0, 2o_{L_j})$	Tasoitusparametri L-suunnan liikkeelle kaarteessa j
$\kappa(x_i)$	Pisteen x_i kaarevuus
$\max \kappa_j$	Maksimi kaarevuus kaarteessa j
$w_j \in \{-1, 1\}$	Liikkumissuuntapainotus kaarteelle j
$O_D(x_i)$	Liikkuma-arvo pisteelle x_i suuntaan D
$O_L(x_i)$	Liikkuma-arvo pisteelle x_i suuntaan L
S	Sallittu alue

6 Simulaatiotulokset

Tässä luvussa esitellään joen tulevaisuutta ennustavan mallinnusmenetelmän tuloksia. Menetelmää sovelletaan kahteen meanderoivaan jokeen, Oulankajokeen ja Pulmankijokeen.

Alla olevissa simulaatioissa algoritmin parametrit on asetettu niin, että ensin migraatio tapahtuu ainoastaan virtaussuuntaisesti, seuraavissa tapauksissa ainoastaan sivuttaissuuntaisesti ja näiden jälkeen joen kaarteet voivat liikkua kumpaankin suuntaan. Lisäksi loppuun on tehty simulaatiot, joissa maksimiliikkumat o_{D_j} ja o_{L_j} voivat saada myös negatiivisia arvoja.

Oulankajoen simulaatioissa on suoritettu 20 iteraatiokierrosta, joissa kaarteiden liikkeiden maksimiarvot noudattavat normaalijakaumaa $N(20, 5)$. Uuden juoluan syntymiseksi algoritmiin on asetettu ehto, että käänne- ja huippupisteiden täytyy olla alle 100 metrin päässä toisistaan. Pulmankijoen simulaatioissa on tehty 30 iteraatiokierrosta käyttäen normaalijakaumaa $N(7, 2)$. Tässä uuden juoluan syntymiselle simulaatioihin on asetettu rajaksi 60 metriä. Parametrit on asetettu siten, että ne soveltuvat tutkimusalueiden mittakaavaan.

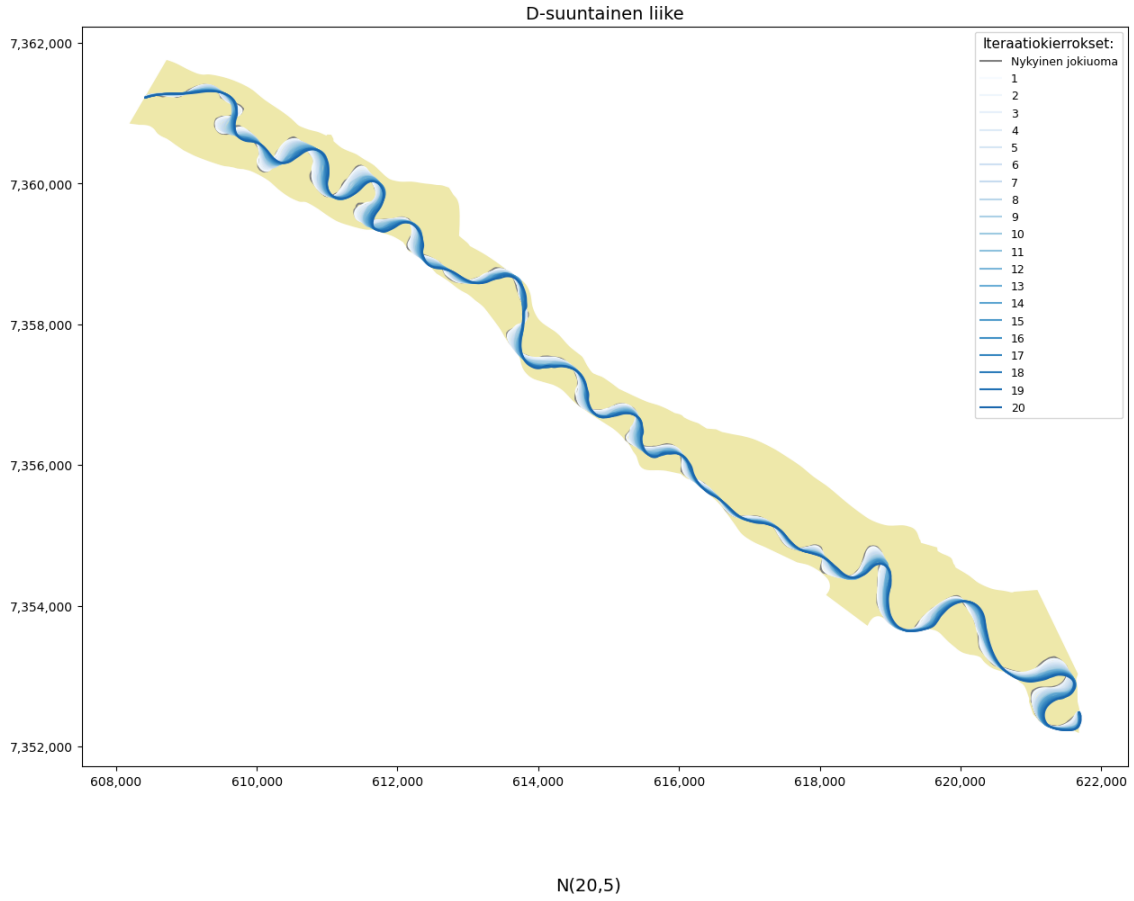
Algoritmin tulokset esitetään joen keskiviivan liikkeenä, havainnollistaen vaaleasta tummaan muuttuvalla värityksellä sitä miten joen keskiviiva kehittyy. Taustalla oleva vaalea alue on maaperäaineistosta muodostettu jokilaakso, joka toimii sallittuna alueena pisteiden liikkumiselle. Jokiuoman kokonaispituus ja sinuositeetti on laskettu simulaation jälkeen viimeiselle keskiviivalle tarkastelemaan kehitystä.

Luvun lopussa esitetään Oulankajoen simulaatioita ilman sallittua aluetta. Tämän avulla voidaan nähdä, miten rajoittavan alueen poistaminen vaikuttaa algoritmin tuloksiin.

Lisäksi tarkastellaan mallinnusmenetelmän satunnaisuuden aiheuttamia eroja simulaatiotuloksiin. Satunnaisuuden vaikutusta havainnollistetaan heatmap-kuvalla, joka esittää, kuinka jokiuoma kehitys vaihtelee 50 erillisessä simulaatiossa, jotka on toteutettu samoilla lähtöarvoilla.

6.1 Virtaussuuntainen migraatio

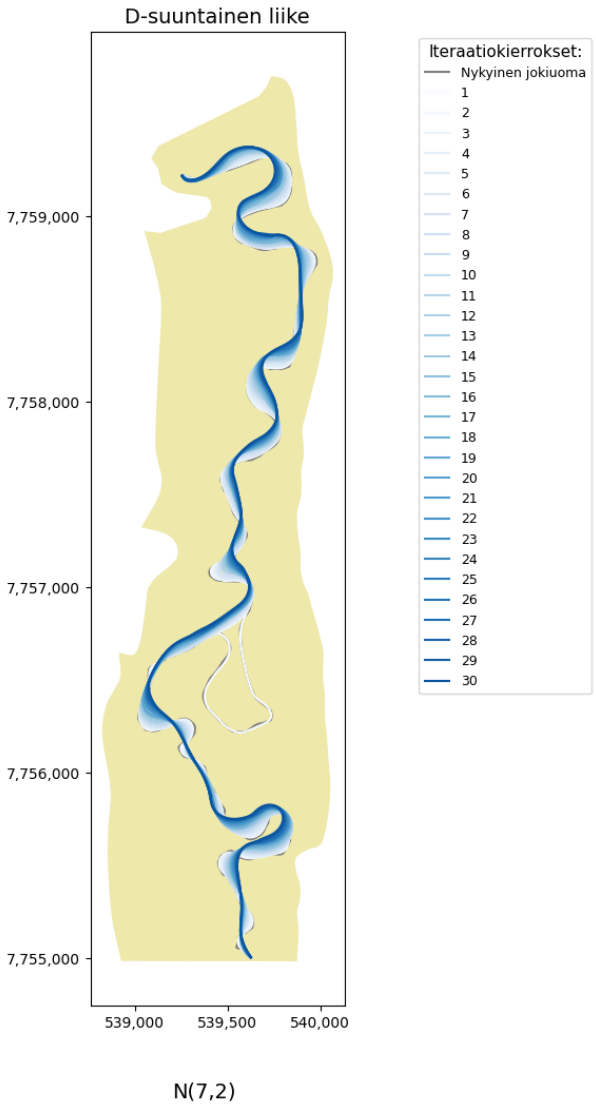
Puhtaassa virtaussuuntaan D tapahtuvassa mallinnuksessa sivuttainen liike L määritellään nollaksi. Tällä simulaatiolla mallinnetaan tilannetta, jossa meanderikaarteet liikkuvat vain kohti joen alajuoksua.



Kuva 31: Puhdas virtaussuuntaan tapahtuvan migraation simulaatio Oulankajoelle. Kaarteet liikkuvat tasaisesti virtaussuunnassa kohti alajuoksua. Suurin virtaussuuntainen migraatio nähdään kaarteiden huipuissa, joihin mallin mukaisesti kohdistuu suurin eroosio.

Taulukko 5: Oulankajoen nykyisen ja ennustetun jokiuoman ominaisuuksien vertailu virtaussuuntaisen migraation jälkeen.

	Nykyinen jokiuoma	Viimeisin ennustettu jokiuoma
Jokiuoman pituus (km)	24.5	21.6
Suoran pituus (km)	15.9	16
Sinuositeetti	1.5	1.4



Kuva 32: Puhdas virtaussuuntaan tapahtuvan migraation simulaatio Pulmankijoelle. Kaarteet liikkuvat virtaussuuntaisesti ja huipussa tapahtuva suurin virtaussuuntainen eroosio vaikuttaa suoristavan jokiuomaa. Kaarteiden virtaussuuntainen liike aiheuttaa uoman katkeamisen ja uuden juoluan syntymisen. Juolua näkyy vaaleana kaarteena kuvassa, kun jokiuoman uusi reitti muodostuu sen ohi.

Taulukko 6: Pulmankijoen nykyisen ja ennustetun jokiuoman ominaisuuksien vertailu virtaussuuntaisen migraation jälkeen.

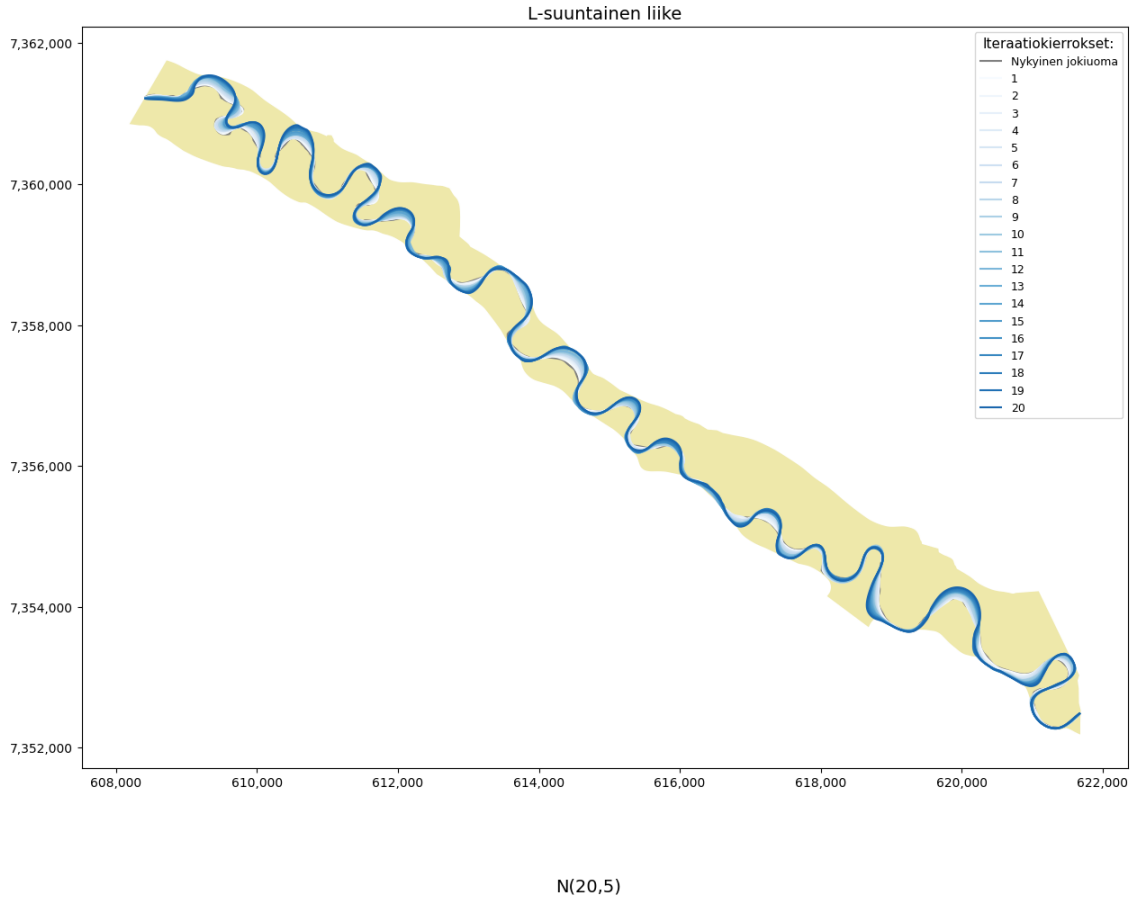
	Nykyinen jokiuoma	Viimeisin ennustettu jokiuoma
Jokiuoman pituus (km)	9.2	6.3
Suoran pituus (km)	4.3	4.4
Sinuositeetti	2.1	1.4

Yllä olevissa simulaatioissa (kuvat 31 ja 32) migraatiomallin mukaisesti suurin liike tapahtuu kaarteiden huipuille virtaussuunnassa. Huippujen migraatio johtaa kaarteiden venymiseen alajuoksun suuntaan. Pulmankijoen simulaatiossa nähdään juoluan syntyminen, kun vierekkäiset kaarteet kuroutuvat kiinni toisiinsa simulaation alussa. Oulankajoen simulaatioissa osa kaarteista liikkuu sallitun alueen reunalle, mikä rajoittaa kyseisten kaarteiden liikettä.

Taulukoista 5 ja 6 nähdään, että simulaatioiden tuottamat viimeisimmät jokiuomat ovat lyhyempiä alkuperäisiin verrattuna. Oulankajoen sinuositeetti laskee simulaation tuloksena arvosta 1.5 arvoon 1.4 ja Pulmankijoen sinuositeetti laskee arvosta 2.1 arvon 1.4. Tulokset viittaavat jokiuomien suoristumiseen, ja taulukon 1 mukaan jokiuomat kuuluvatkin simulaatioiden jälkeen ”Loivasti meanderoiva” -luokkaan. Pulmankijoen tunnuslukujen suurempiin muutoksiin vaikuttaa meanderikaarteen irtoaminen simulaation alussa, mikä lyhentää jokiuomaa huomattavasti.

6.2 Sivuttaissuuntainen migraatio

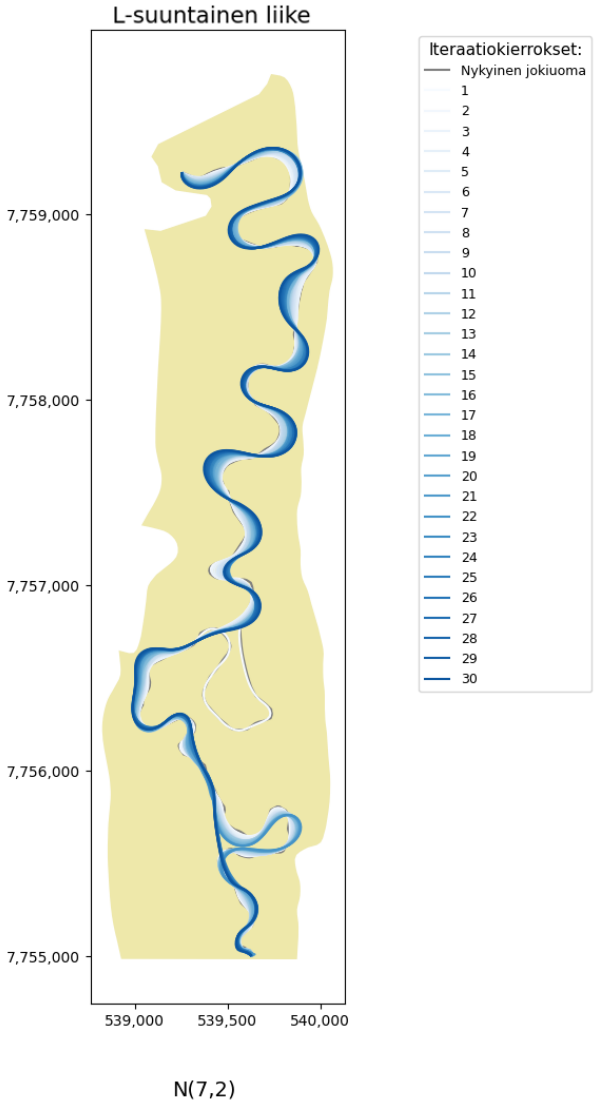
Puhtaan sivuttaissuuntaisen liikkeen L tapauksessa, migraatio tapahtuu vain meanderikaarteiden ulkoreunaan kohti. Tässä tapauksessa virtaussuuntaan tapahtuva liike D on määritelty nollaksi.



Kuva 33: Puhdas sivuttaissuuntaan tapahtuva migraatio Oulankajoelle. Huippuihin kohdistuva migraatio venyttää kaarteita sivuttaissuuntaan, ja usean kaarteen nähdään siirtyvät sallitun alueen reunalle.

Taulukko 7: Oulankajoen nykyisen ja ennustetun jokiuoman ominaisuuksien vertailu sivuttaissuuntaisen migraation jälkeen.

	Nykyinen jokiuoma	Viimeisin ennustettu jokiuoma
Jokiuoman pituus (km)	24.5	28.4
Suoran pituus (km)	15.9	15.9
Sinuositeetti	1.5	1.8



Kuva 34: Puhdas sivuttaissuuntaan tapahtuva migraatio Pulmankijoelle. Kaarteiden leveneminen huppuja kohti aiheuttaa uoman katkeamisen kahdessa kohtaa jokiuomaa. Uudet juoluat nähdään kuvassa vaaleiksi jääneinä alueina, kun jokiuoman uusi reitti muodostuu niiden ohi.

Taulukko 8: Pulmankijoen nykyisen ja ennustetun jokiuoman ominaisuuksien vertailu sivuttaissuuntaisen migraation jälkeen.

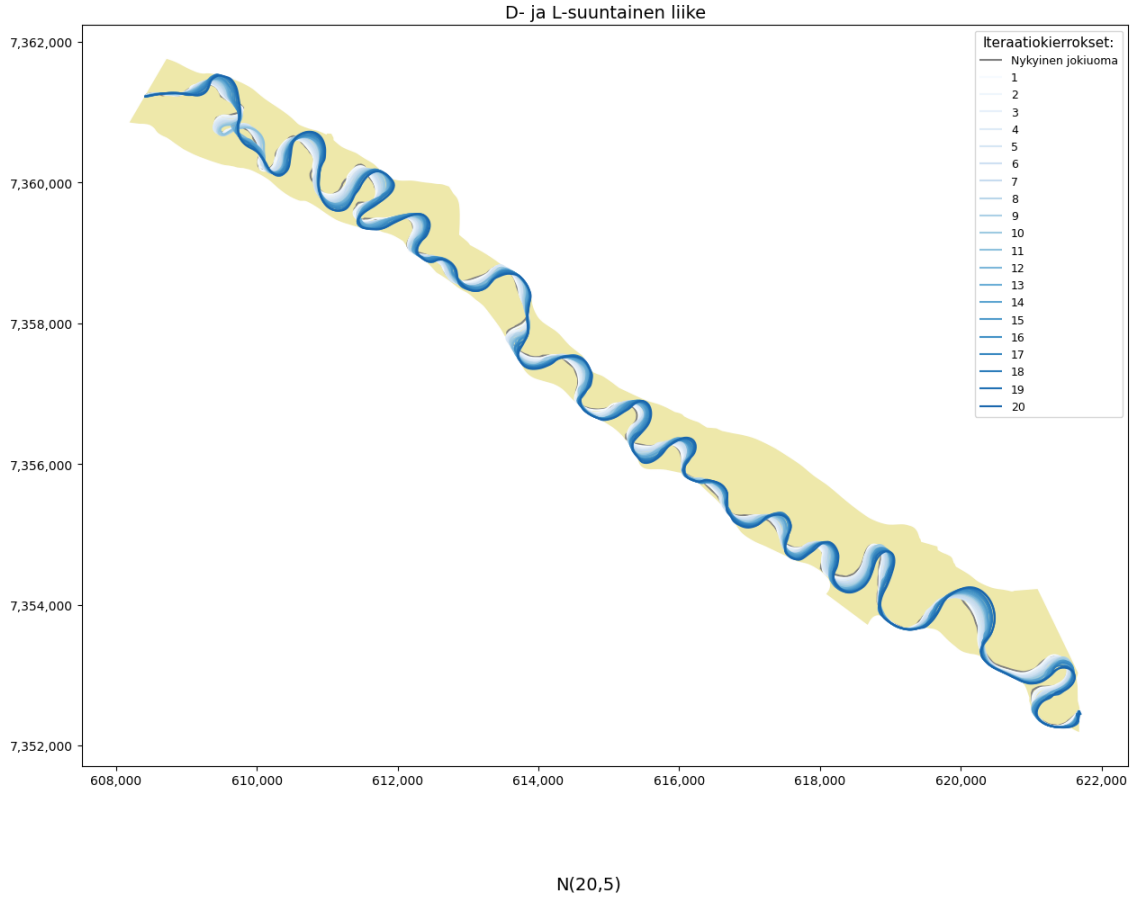
	Nykyinen jokiuoma	Viimeisin ennustettu jokiuoma
Jokiuoman pituus (km)	9.2	8.1
Suoran pituus (km)	4.3	4.4
Sinuositeetti	2.1	1.9

Puhtaan sivuttaisen migraation tapauksissa jokiuoma kaarteiden nähdään laajentuvan (kuvat 33 ja 34). Sallittu alue rajoittaa migraatiota selvästi enemmän, kuin virtaussuuntaan tapahtuvassa simulaatioissa. Erityisesti Oulankajoen simulaatioissa useat meanderikaarteen liikkuvat sallitun alueen reunalle asti, minkä jälkeen kaarteet enää hieman liikkuvat sallitun alueen reunaa pitkin. Pulmankijoen simulaatiossa havaitaan kahden uuden juoluan muodostuvan, kun kaarteet levenevät kiinni toisiinsa.

Taulukosta 7 nähdään että Oulankajoen kaarteiden leveneminen aiheuttaa jokiuoman kokonaispituuden kasvamisen noin neljällä kilometrillä ja näin myös sinuositeetti kasvaa arvosta 1.5 arvoon 1.8. Taulukosta 8 nähdään, että Pulmankijoen kokonaispituus laskee noin kilometrillä ja sinuositeetti laskee arvosta 2.1 arvoon 1.9. Vaikka tunnusluvut osoittavat laskua, muiden kaarteiden leveneminen kompensoi muutoksia niin, etteivät luvut laske merkittävästi kahden suurikokoisen kaarteen irtoamisesta huolimatta.

6.3 Virtaus- ja sivuttaissuuntainen migraatio

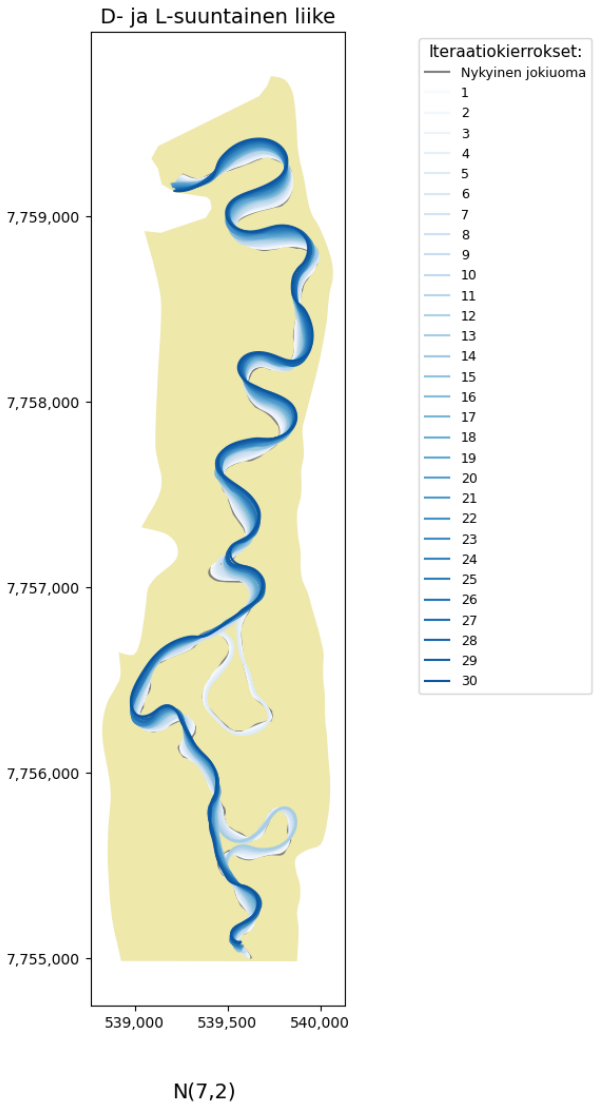
Sekoitetussa tapauksessa meanderikaarteiden migraatio tapahtuu sekä virtaussuuntaan D että sivuttaissuuntaan L.



Kuva 35: Virtaus- ja sivuttaissuuntaan tapahtuva migraatio Oulankajoelle. Nähdään, että nyt kaarteet samanaikaisesti sekä levenevät että siirtyvät alajuoksua kohti, mikä tekee jokiuoman liikkeestä kokonaisvaltaisempaa. Sallitun alueen havaitaan rajoittavan useiden kaarteiden migraatiota. Lisäksi jokiuoman alkuosassa nähdään muodostuvan kaksi uutta juoluaa.

Taulukko 9: Oulankajoen nykyisen ja ennustetun jokiuoman ominaisuuksien vertailu virtaus- ja sivuttaissuuntaisen migraation jälkeen.

	Nykyinen jokiuoma	Viimeisin ennustettu jokiuoma
Jokiuoman pituus (km)	24.5	28.8
Suoran pituus (km)	15.9	15.9
Sinuositeetti	1.5	1.8



Kuva 36: Virtaus- ja sivuttaissuuntaan tapahtuva migraatio Pulmankijoelle. Kaarteiden nähdään sekä levenevän että liikkuvan virtaussuunnan mukaisesti. Kahdessa kohtaa simulaatiota jokiuoma oikaisee reitin ja muodostaa uuden juoluan.

Taulukko 10: Pulmankijoen nykyisen jokiuoman ja ennustetun jokiuoman ominaisuuksien vertailu.

	Nykyinen jokiuoma	Viimeisin ennustettu jokiuoma
Jokiuoman pituus (km)	9.2	7.6
Suoran pituus (km)	4.3	4.3
Sinuositeetti	2.1	1.7

Molemmat liikkumissuunnat sallivissa simulaatioissa havaitaan pinta-alallisesti suurin migraatio jokiuomille (kuvat 35 ja 36). Meanderikaarteiden nähdään samanaikaisesti vaeltavan virtaussuuntaan sekä levenevän sivuttaissuuntaan. Jokiuomat näyttävät säilyttävän alkuperäistä muotoaan eniten verrattuna aiempiin simulaatioihin, sillä kaarteiden liike on nyt tasaisempaa, kun se tapahtuu molempiin suuntiin.

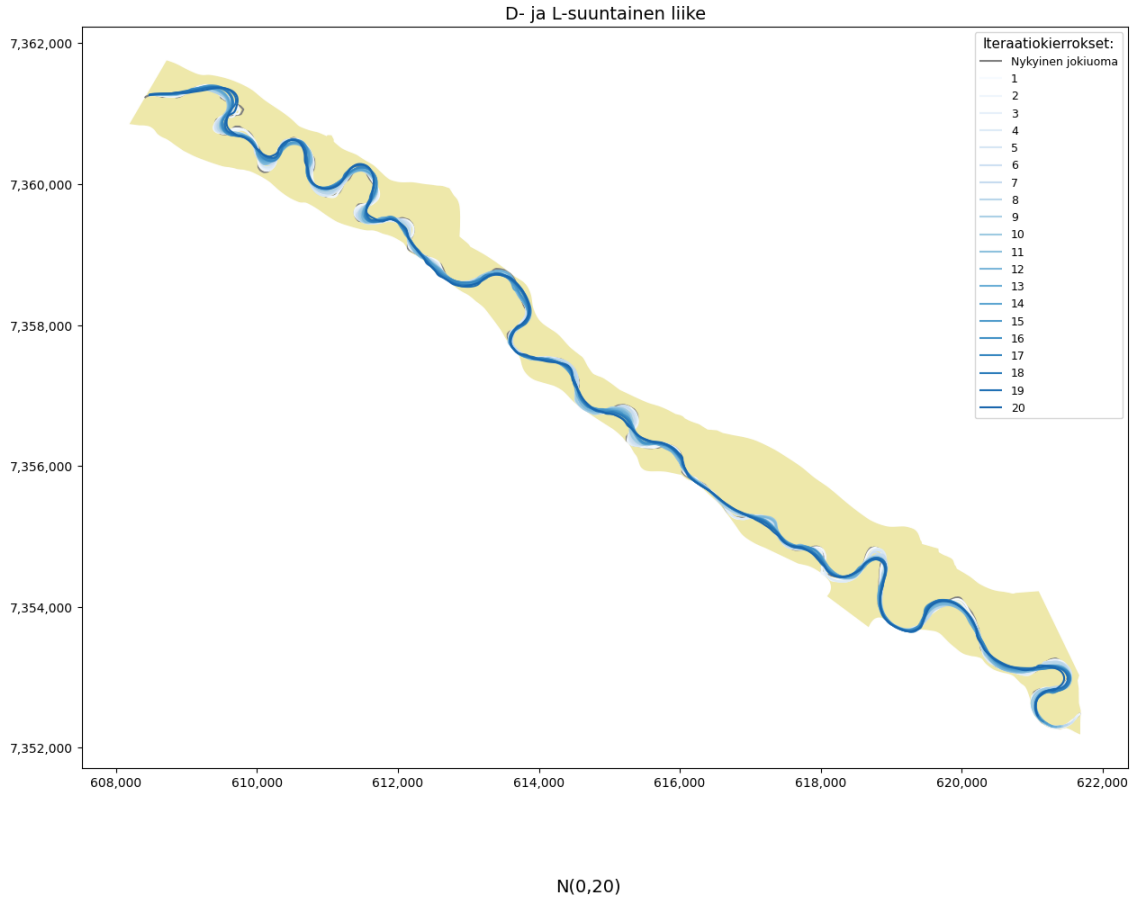
Molempien jokiuomien simulaatioissa nähdään uusien juoluoiden muodostuminen. Oulankajoen alkuosassa muodostuu kaksi uutta juoluaa, ja Pulmankijoen simulaatiossa syntyy samat juoluat kuin sivuttaissuuntaisen migraation simulaatiossa (kuva 34). Pulmankijoen alkuosan juolua muodostuu nyt kuitenkin hieman aikaisemmin, mikä saattaa johtua virtaussuuntaisesta liikkeestä, joka voi nopeuttaa kaarteiden lähestymistä.

Taulukosta 9 nähdään Oulankajoen jokiuoman pituuden kasvaneen simulaation jälkeen yli neljä kilometriä ja sinuositeetin nousseen arvoon 1.8. Muutokset luvuissa ovat melko samat kuin sivuttaissuuntaisen migraation tuloksissa, vaikka kaksi kaarretta on nyt leikkautunut pois jokiuomasta. Tämä viittaa siihen, että uoma on kasvanut pituudeltaan ja kaarevuudeltaan hieman enemmän tässä simulaatiossa.

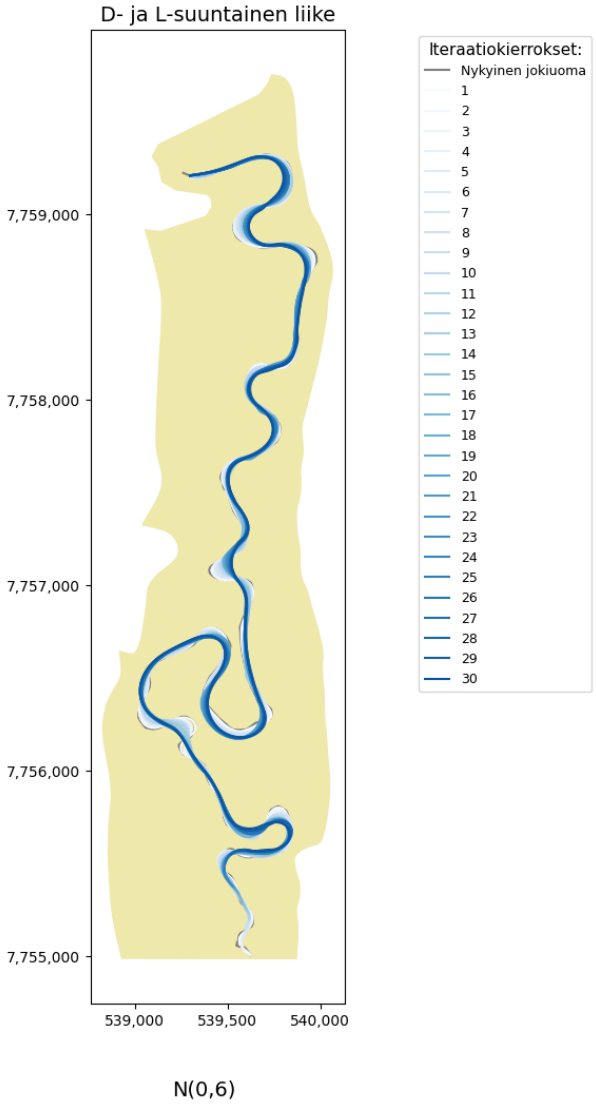
Pulmankijoen tunnuslukutaulukosta 10 nähdään, että uoman kokonaispituus on laskenut noin 1.5 kilometriä ja sinuositeetti on laskenut arvoon 1.7. Lukujen laskua selittää kahden suuren juoluan irtoaminen simulaation aikana. Jäljelle jäävän jokiuoman pituuden ja kaareilevuuden kasvu tasoittaa juoluoiden irtoamisen aiheuttamaa vaikutusta, minkä vuoksi tunnuslukujen muutokset jäävät melko vähäisiksi.

6.4 Molempiin suuntiin tapahtuva migraatio ja negatiivinen maksimiliikkuma

Alla olevissa simulaatioissa migraatio on määritelty tapahtuvan molempiin suuntiin D ja L. Maksimiliikkuma-arvot o_D ja o_L otetaan nyt Oulankajoelle normaalijakaumasta $N(0, 20)$ ja Pulmankijoelle normaalijakaumasta $N(0, 6)$. Tällöin siis kaarteelle maksimiliikkumaksi molemmille tai toiselle suunnalle voi määräytyä myös negatiivinen arvo, jolloin kaarteen pisteet voivat liikkua taaksepäin.



Kuva 37: Virtaus- ja sivuttaissuuntaan tapahtuva migraatio Oulankajoelle. Nähdään, että negatiivisen liikkeen mahdollisuus aiheuttaa edestakaista liikettä kaarteissa, mikä vähentää jokiuoman etenemistä. Vähäinen muutos voi johtua myös pienistä liikkumisarvoista. Kärjissä tapahtuva suurin liike vaikuttaa suoristavan kaarteita ja loppuosassa jokea hieman kääntävän kaarteita.

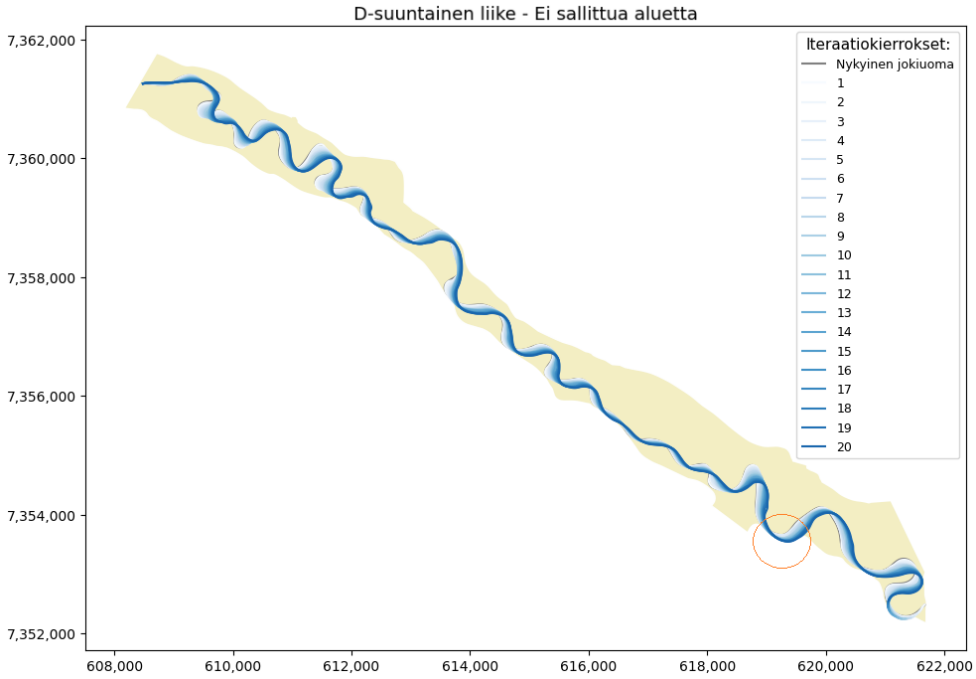


Kuva 38: Virtaussuuntaan ja sivuttaissuuntaan tapahtuva migraatio Pulmankijoelle. Edestakainen ja vähäinen liike hidastaa myös Pulmankijoen migraatiota ja suoristaa osaa sen kaarteista.

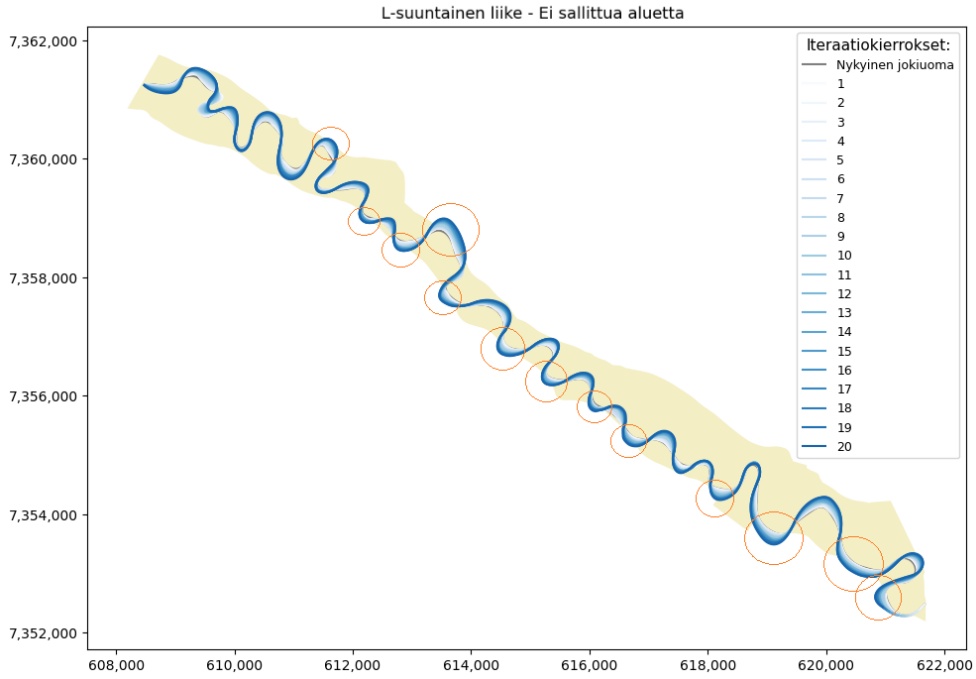
Verrattuna edellisiin simulaatioihin, yllä olevissa tapauksissa taaksepäin suuntautuva liike tapahtuu herkemmin. Simulaatiokuvien perusteella tämä näyttää pääasiassa hidastavan uoman migraation etenemistä. Kaarteiden suorilla osuuksilla liikettä esiintyy vain vähän ja kaarteiden huiput useissa kohdissa suoristuvat edestakaisen liikkeen seurauksena.

6.5 Simulaatio ilman sallittua aluetta

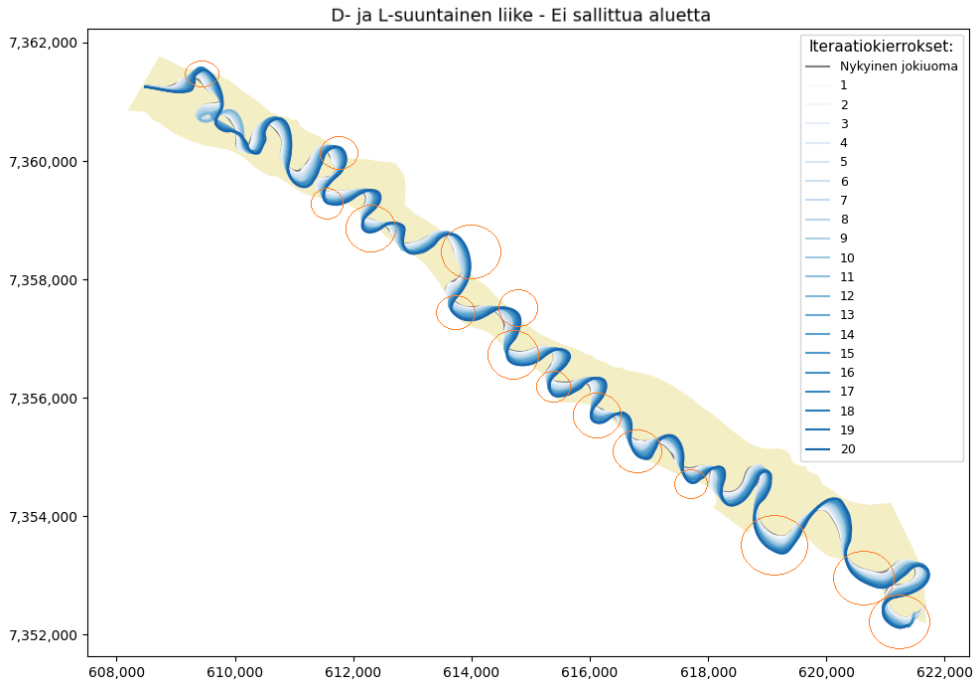
Oulankajoen simulaatioissa havaittiin, että jokiuoma kulkeutui lähelle sallitun alueen reunoja, jolloin alue rajoittaa jokiuoman migraatiota. Alla olevissa kuvissa esitetään simulaatio tilanteessa, jossa rajoittavaa aluetta ei ole käytetty. Tällöin jokiuoma voi liikkua vapaasti myös sallitun alueen ulkopuolelle. Tämä havainnollistaa rajoittavan alueen merkitystä jokiuoman liikkeen ja simulaation lopputuloksen kannalta. Kuvissa nähdään sallittu alue taustalla ja oranssilla ympyröitynä kaarteet, jotka ovat ylittäneet alueen.



Kuva 39: Simulaatio ilman rajoittavaa aluetta D-suuntaisella liikkeellä. Tässä simulaatiotapauksessa yhden kaarteen nähdään ylittävän sallitun alueen. Kaarteet liikkuvat jokilaakson suuntaisesti, jolloin ne pysyvät hyvin alueen sisäpuolella. Muutamien kaarteiden nähdään liikkuvan lähelle alueen reunoja, joten jos simulaatiota jatkettaisiin pidempää, voisi useampi kaarre ylittää alueen.



Kuva 40: Simulaatio ilman rajoittavaa aluetta L-suuntaisella liikkeellä. Nähdään että useampi kaarre ylittää sallitun alueen, kun kaarteet levenevät.



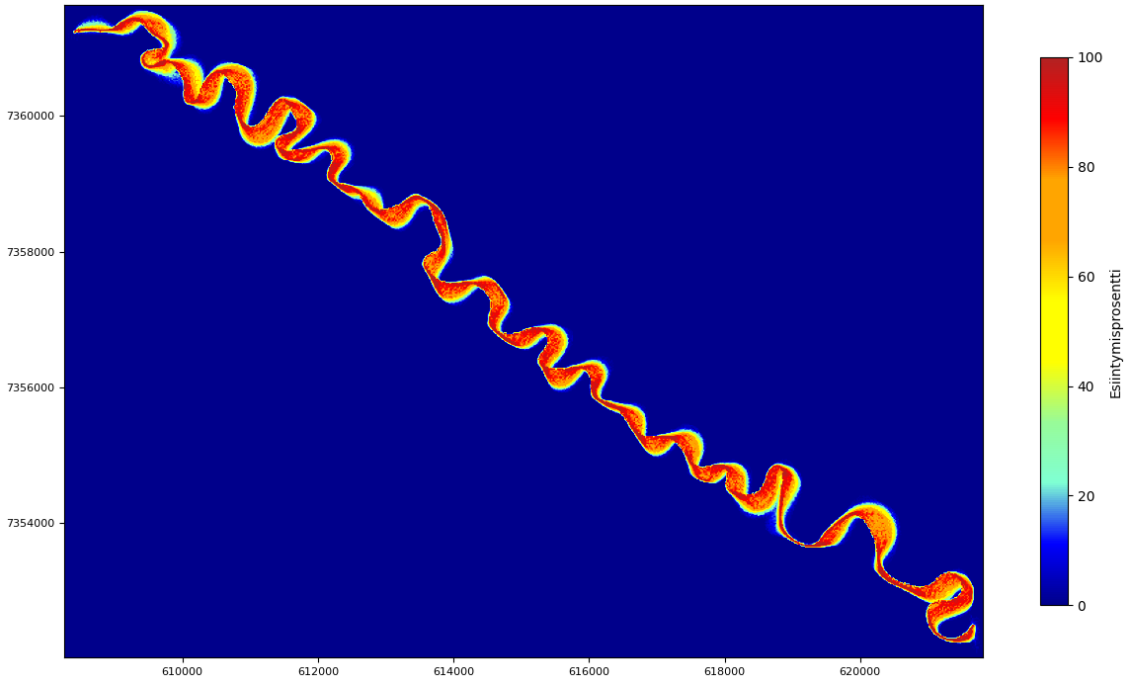
Kuva 41: Simulaatio ilman rajoittavaa aluetta D- ja L-suuntaisella liikkeellä. Nähdään että useampi kaarre ylittää sallitun alueen, kun kaarteet levenevät ja liikkuvat virtaussuuntaisesti.

Vertailu osoittaa, että kaarteet liikkuvat huomattavasti vapaammin, kun rajoittavaa aluetta ei käytetä Oulankajoen simulaatiossa. Molemmat suunnat sallivassa tapauksessa (kuva 41) sekä pelkän sivuttaissuuntaisen liikkeen L tapauksessa (kuva 40) moni kaarre ylittää sallitun alueen, kun jokiuoma saa liikkua vapaasti. Pelkän virtaussuunnan D sallivassa tapauksessa (kuva 39) vain yksi kaarre ylittää sallitun alueen.

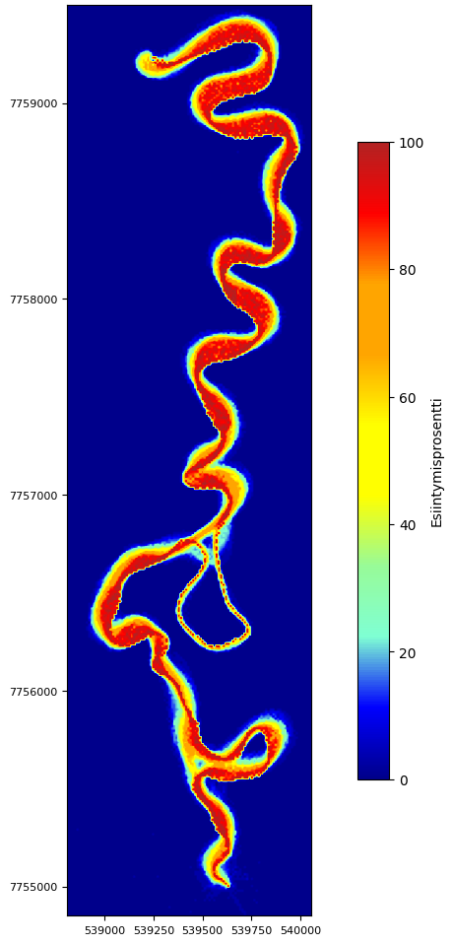
6.6 Satunnaisuuden vaikutus

Mallinnusmenetelmässä hyödynnetään stokastisuutta, minkä seurauksena jokaisen simulaation tulos on erilainen. Alla esitettävissä visualisoinneissa havainnollistetaan satunnaisuuden vaikutusta lopputuloksiin.

Luvun 6.3 mukaisia molemmat liikkumasuunnat sallivia simulaatioita on suoritettu 50 molemmille jokiuomille. Näiden tuloksista on laadittu heatmap-kuvat, jotka havainnollistavat prosentuaalisesti jokiuomien sijainnin ja muodon vaihtelua simulaatioiden välillä.



Kuva 42: Heatmap-kuva Oulankajoelle 50 kertaa toistetusta simulaatiosta. Punaiset alueet kuvaa jokiuoman yleisimpiä sijainteja sekä alkuperäistä jokiuomaa, kun taas sinisillä ja vihreillä alueilla jokiuoma esiintyi harvemmin. Keltaisille alueille jokiuoma päätyi noin puolissa simulaatioissa.



Kuva 43: Heatmap-kuva Pulmankijoelle 50 kertaa toistetusta simulaatiosta. Punaiset alueet kuvaavat jokiuoman yleisimpiä sijainteja sekä alkuperäistä jokiuomaa, kun taas sinisillä ja vihreillä alueilla jokiuoma esiintyi harvemmin. Keltaisille alueille jokiuoma päätyi noin puolissa simulaatioissa.

Oulankajoen heatmap-kuvasta (kuva 42) voidaan havaita, että jokiuoma kehittyy suhteellisen samanlaisesti kaikissa simulaatioissa. Kaarteiden päissä nähtävät punaisesta poikkeavat alueet viittaavat siihen, että joissain simulaatioissa jokiuoma kehittyy hieman kauemmas tai eri suuntaan näillä alueilla. Oulankajoen alkuosan pienen kaarteen kohdalla on havaittavissa punertava alue, joka menee kaarteen läpi. Tämä voisi viitata simulaationkuvan (kuva 35) tapaiseen juoluan muodostumiseen tai siihen, että viereinen kaarre venyy kyseiselle alueelle.

Pulmankijoen heatmap-kuvassa (kuva 43) juoluoiden muodostumiset luo enemmän eroja kehityspolkujen välillä eri simulaatioissa. Muiden kaarteiden kehitykset näyttävät suhteellisen yhtenäiseltä, lukuun ottamatta kaarteiden huippuja, jotka osassa simulaatioita siirtyvät hieman kauemmaksi. Pulmankijoen alkupäässä nähdään kaarteen irtoavan jokiuomasta yli puolessa simulaatioista. Seuraava irtoava

kaarre puolestaan leikkautuu yli 60 %:ssa simulaatioista, ja sillä havaitaan olevan myös toinen mahdollinen irtoamiskohta, joka esiintyy noin 30 %:ssa tapauksista. Näiden havaintojen perusteella voidaan olettaa, että kyseinen kaarre irtoaa lähes kaikissa simulaatioissa.

6.7 Algoritmin laskenta-aika

Alla olevissa taulukoissa esitetään algoritmin suorittamiseen kuluneet laskenta-ajat (CPU) molemmat liikkumissuunnat sallivien simulaatiotulosten tuottamiseen eri lähtöarvoilla. Simulaatiot suoritettiin Dell Latitude 5450 -tietokoneella, jossa on Intel Core Ultra 7 -prosessori (3.8 GHz) ja 32 GB RAM -muisti.

Taulukko 11: Algoritmin laskenta-ajat eri lähtöarvoilla Oulankajoelle

Liikkumamäärät o_{D_j} ja o_{L_j}	N_s	CPU aika (s)
N(20,5)	15	2.81
N(20,5)	20	3.93
N(30,7)	20	4.01
N(20,5)	25	4.85
N(30,7)	25	5.08

Taulukko 12: Algoritmin laskenta-ajat eri lähtöarvoilla Pulmankijoelle

Liikkumamäärät o_{D_j} ja o_{L_j}	N_s	CPU aika (s)
N(7,2)	20	1.91
N(7,2)	30	2.85
N(10,3)	30	2.84
N(7,2)	40	3.82
N(10,3)	40	3.95

Taulukkojen 11 ja 12 mukaan suoritusaika näyttää kasvavan lähes lineaarisesti suhteessa simulaatiokierrosten lukumäärän N_s. Suurta vaikutusta laskenta-aikaan ei näytä olevan eri o_{D_j} ja o_{L_j} asetuksilla, jos kierrosmäärät pysyvät samana. Oulankajoen laskenta-aikojen pidempi kesto selittyy tutkimusalueen suuremmalla koolla.

7 Pohdinta

Tutkielmassa kehitetyllä mallinnusmenetelmällä onnistuttiin simuloimaan Oulankajoen ja Pulmankijoen jokiuomien tulevaa migraatiota sekä uusien juoluoiden muodostumista. Puhtaan sivuttaissuuntaisen migraation simulaatiotuloksissa nähtiin jokiuoman kaarteiden levenevän ja näin kasvattavan jokiuoman pituutta sekä sinuositeettia. Puhtaan virtaussuuntaisen migraation simulaatioilla onnistuttiin mallintamaan jokiuoman siirtymistä kohti alajuoksua, ja molemmat liikkumasuunnat sallivilla simulaatioilla onnistuttiin siirtämään kaarteita kohti niiden huippuja sekä virtaussuuntaan. Lisäksi sallitulla alueella onnistuttiin rajaamaan jokiuoman muutoksia, mikä paransi tulosten realistisuutta ja vastasi tutkielman tavoitteita.

Parquer ym. [21] kehittämässä ChaRMigS-menetelmässä voidaan jokiuoman migraatiota rajoittaa sallimalla kaarteiden liike ainoastaan ylävirtaa kohti tai leveyssuunnassa. Uudessa lähestymistavassa sallitulla alueella pystytään rajoittamaan liikettä ympäristön asettamin rajoin. Tämä parantaa menetelmän sovellettavuutta jokiuomiin, joiden migraatioalue on rajoittunut esimerkiksi kallioperän muotojen vuoksi, mutta silti säilytetään mahdollisuus liikuttaa kaarteita monensuuntaisesti. Tutkielmassa tehdyissä Oulankajoen simulaatioissa nähdään, että sallitun alueen käyttöönotto rajoitti jokiuoman liikettä ja sillä oli selkeä vaikutus lopputulokseen. Oulankajoen kaarteiden nähtiin selkeästi laajenevan enemmän ilman rajoittavaa aluetta tehdyissä simulaatioissa (kuva 41), kuin rajoittavan alueen simulaatiossa (kuva 35), joissa rajalle osuvien kaarteiden huiput levenevät reunaa pitkin. Näin ollen simulaation tuloksesta saatiin realistisempi ympäristön rajoitteet huomioon ottaen. Pulmankijoen kohdalla sallittu alue on sen verran leveä, ettei sillä ollut paljoa vaikutusta simulaatioon. Jos kuitenkin simulaatioita jatkettaisiin pidemmälle, voitaisiin siinäkin päästä sallitun alueen reunoille.

Sallitun alueen osalta algoritmia voisi kuitenkin kehittää siten, että jos piste päätyy sallitun alueen ulkopuolelle, piste siirrettäisiin lähimmälle reunapisteelle sen sijaan, että pistettä ei siirretä ollenkaan. Myös molempiin suuntiin tapahtuvan liikkeen tapauksessa, jos piste ylittää sallitun alueen esimerkiksi L suunnassa niin voitaisiin sen silti antaa liikkua D suunnassa. Tällöin jokiuoma saattaisi liikkua sallitun alueen reunaa pitkin nopeammin kuin nykyisessä versiossa. Nyt tämä tapahtuu vain jos seuraavien kierrosten liikkumisarvot ja -suunnat sattuvat olemaan sopivat.

Algoritmille voidaan syöttää eri parametreja todennäköisyysjakaumaan, jolla määritetään kaarteiden maksimiliikkumat. Parametreja säätämällä voidaan toteuttaa erikokoisia simulaatioita. Stokastisuuden käyttö tuo mallinnusmenetelmään luonnollista vaihtelevuutta. Tällöin menetelmä ei myöskään edellytä tarkkaa tietoa jokiuoman käyttäytymisestä ja se mahdollistaa simuloinnin myös silloin, kun lähtö-

tiedot ovat rajalliset.

Nykyisessä menetelmässä maksimiliikkumat määrätään kaikille kaarteille samojen parametrien perusteella, mikä johtaa siihen, että suuresti liikkuvien kaarteiden ennusteet sulautuvat muiden joukkoon ja menetelmän tarkkuus heikkenee. Tarkkuutta voitaisiin parantaa mahdollistamalla todennäköisyysjakauman parametrien määrittäminen yksilöllisesti kaarteille. Näin otettaisiin huomioon, että jokiuoman sisällä kaarteet saattavat käyttäytyä eri tavoin [10, 14, 21].

Luvun 6.4 simulaatioissa maksimiliikkumat perustuvat todennäköisyysjakaumaan, jonka odotusarvona on nolla. Tällöin sekä negatiivinen että positiivinen liike on yhtä todennäköistä, jolloin kaarteiden migraatioiden voisi olettaa olevan monimutkaisempia, kun pisteet voivat liikkua edestakaisin. Simulaatiotulokset osoittivat kuitenkin tämän pääosin vain hidastavan ja vähentävän migraatiota. Nollan odotusarvo johtaa siihen, että liikkuma-arvot ovat useimmissa tapauksissa pieniä, mikä vähentää liikkeen määrää. Mikäli osattaisiin määritellä ennalta kaarteet, joille edestakainen liike on ominaista, voitaisiin normaalijakaumaa odotusarvolla nolla käyttää näille kaarteille. Muille kaarteille liikkumat voitaisiin puolestaan määrittää positiivisen luvun antavasta jakaumasta. Odotusarvoa voisi mahdollisesti myös kasvattaa hieman tai käyttää jotain muuta todennäköisyysjakaumaa, jotta tulos olisi mielekkäämpi.

Alkuperäinen ChaRMigS-menetelmä perustuu Missisippijoen tutkimuksesta saatuihin korrelaatiotuloksiin, joissa kaarteiden migraatiota verrattiin normalisoituihin kaarevuusarvoihin. Tulokset osoittivat vaihtelua siinä tapahtuiko liikettä enemmän virtaus- vai sivuttaissuuntaan kaarevuuden kasvaessa. Mallinnusmenetelmä perustuu tähän tulokseen ja näin ollen kaarteiden liike voi tapahtua moneen suuntaan. Liikkeen monipuolisuus mahdollistaa mallin käytön erilaisille jokiuomille ja tekee siitä käyttökelpoisen myös tulevaisuuden mallintamiseen.

Algoritmissa määrätään satunnaisesti valitulla liikkumissuunnan painotusparametrilla w onko kaarteen muutos suurempaan virtaus- vai sivuttaissuuntaan. Parametri w valitaan jokaisella kierroksella uudelleen, jolloin suunnanpainotus voi muuttua. Algoritmin tarkkuutta voitaisiin parantaa määräämällä osalle kaarteista suuntapainotusparametri w vakioksi, jos kyseiselle kaarteelle tiedetään suunta, johon se liikkuu enemmän. Lisäksi, jos jokiuomalle tiedettäisiin jokin yleinen ominaisuus, jonka perusteella sen kaarteet liikkuvat enemmän virtaus- tai sivuttaissuuntaan, voitaisiin tämä lisätä algoritmiin rajoittamaan parametria w.

Kaarteen pisteitä siirtävät mallit (5) ja (6) liikuttavat eniten kaarteiden huippuja, jolloin migraatio on suurinta huippujen kohdalla. Sivuttaissuuntaisessa migraatiossa kaarteiden huiput venyvät leveyssuunnassa kaikkein voimakkaimmin, ja virtaus-

suuntainen migraatio puolestaan siirtää huippuja eniten virtaussuuntaan. Näin käy siitä huolimatta vaikka kaarteen luonnollinen migraatio tapahtuisikin vastakkaiseen suuntaan, kuten esimerkiksi kääntyneiden kaarteiden tapauksessa. Simulaation tulokset eivät siis aina vastaa kaarteiden luonnollista käyttäytymistä ja tätä voitaisiin parantaa määrittelemällä kaarteille yksilölliset migraatiosuunnat kaarteen muotoon ja orientaatioon perustuen.

Molempien jokien simulaatioissa onnistuttiin mallintamaan jokiuoman oikaisuja ja uusien juoluoiden syntymisiä. Juoluan syntymisen ehtona on, että käänne- tai huippupisteet sijaitsevat alle tietyn etäisyyden päässä toisistaan. Nämä pisteet muodostavat usein kapeimmat kohdat, mikä mahdollistaa kaarteiden yhdistymisen. Lisätarkkuutta tähän saataisiin huomioimalla myös muiden pisteiden etäisyydet toisiinsa, sillä kaarteet voivat yhdistyä muistakin kohdista. Käytetty ehto kuitenkin yksinkertaistaa algoritmin laskentaa.

Simulaation ajallisen määrän arviointiin tarvitaan tietoa jokiuoman tyypillisestä migraatiomäärästä ja siksi simulaatiotuloksissa on helpompi puhua migraatiokierroksista kuin ajallisesta määrästä. Jos joelle tiedetään keskimääräinen migraatiomäärä vuodessa, niin kertomalla maksimiliikkumat ja simulaatiokierrokset voitaisiin saada arvio simulaation kokonaisajasta. Esimerkiksi Oulankajoen tapauksessa, kun $o_D \sim N(20, 5)$ ja $o_L \sim N(20, 5)$ ja simulaatioita suoritettiin 20 kierrosta, tämän voidaan ajatella vastaavan noin $300 - 500$ vuoden ajanjaksoa perustuen sen keskimääräiseen noin yhden metrin vuotuiseen migraatioon [12, 13]. Menetelmä sallii liikkuma-arvoille kuitenkin niin paljon muutoksia, että tarkan ajallisen arvion antaminen on vaikeaa.

Tutkielmassa kehitetty mallinnusmenetelmä tarjoaa työkalun meanderoivien jokiuomien migraation ennustamiseen. Menetelmällä voidaan simuloida jokiuomien mutkittelua ja sallitun alueen käyttö mahdollistaa realistisempien tulosten saavuttamisen. Satunnaisuus tuo malliin luonnollista vaihtelua ja mahdollistaa menetelmän käytön myös rajallisten lähtötietojen tilanteissa.

Vaikka menetelmällä on vielä kehityspotentiaalia ja sen antamia tuloksia voitaisiin parantaa säätämällä parametrien arvoja lähemmäs tunnettuja ja yksilöllisiä arvoja, se on jo nyt tuottanut tuloksia jokiuoman muutosten ennustamiselle. Mallin laajentaminen esimerkiksi lisäämällä ympäristötekijöitä huomioivia lisämoduuleja voisi edelleen vahvistaa sen suorituskykyä. Menetelmä tarjoaakin lupaavia mahdollisuuksia meanderoivien jokiuomien pitkän aikavälin muutosten syvällisempään ymmärtämiseen ja ennustamiseen.

Lähteet

[1] Aaltonen, V. T., Aarnio, B., Hyyppä, E., Kaitera, P., Keso, L., Kivinen, E., Kokkonen, P., Kotilainen, M. J., Sauramo, M., Tuorila, P., & Vuorinen, J. (1949). Maaperäsanaston ja maalajien luokituksen tarkistus v. 1949. Agricultural and Food Science, 21(1), 37–66. https://doi.org/10.23986/afsci.71269

[2] Carbonneau, P., & Piégay, H. (2012). Fluvial remote sensing for science and management (1st ed.). Wiley-Blackwell.

[3] Charlton, R. (2008). Fundamentals of fluvial geomorphology. Abingdon; Routledge.

[4] Esri. (2022). ArcGIS Pro: Version 3.0.3. Redlands, CA: Environmental Systems Research Institute. Viitattu 15.12.2024. https://www.esri.com

[5] Geologian tutkimuskeskus. (2024). Hakku. Viitattu 7.12.2024. https://hakku.gtk.fi/

[6] GeoPandas Documentation. (2024). GeoPandas 1.0.1 Documentation. Viitattu 7.12.2025. https://geopandas.org

[7] Harjulehto, P., Klén, R., & Koskenoja, M. (2014). Analyysiä reaaliluvuilla. Gaudeamus Oy.

[8] Hjulström, F. (1935). Studies of morphological activity of rivers as illustrated by the river Fyris. Bulletin of the Geological Institute, University of Uppsala.

[9] Holden, J. (2012). An introduction to physical geography and the environment (3rd ed.). Pearson.

[10] Ielpi, A., & Lapôtre, M. G. A. (2020). A tenfold slowdown in river meander migration driven by plant life. Nature Geoscience, 13(1), 82–86. https://doi.org/10.1038/s41561-019-0491-7

[11] Koppinen, Markku. (2016). Analyyttinen geometria. Turun yliopisto. Viitattu 13.12.2023. https://users.utu.fi/iatorm/wp-content/uploads/sites/1296/2021/10/AnalGeom.pdf

[12] Koutaniemi, L. (1984). The role of ground frost, snow cover, ice break-up and flooding in the fluvial processes of the Oulanka river, NE Finland. Fennia, 162(2).

[13] Koutaniemi, L. (2000). Meanderointi ja sen yhdentoista vuoden seuranta Oulankajoella Kuusamossa. Terra, 112(4).

[14] Liu, C., Liu, A., He, Y., & Chen, Y. (2021). Migration rate of river bends estimated by tree ring analysis for a meandering river in the source region of the Yellow River. International Journal of Sediment Research, 36(5), 593–601. https://doi.org/10.1016/j.ijsrc.2021.04.001

[15] Maanmittauslaitos. (2024). Karttapaikka. Viitattu 7.12.2024. https://www.maanmittauslaitos.fi/asioi-verkossa/karttapaikka

[16] Maanmittauslaitos. (2024). Koordinaattijärjestelmät. Viitattu 20.5.2024. https://www.maanmittauslaitos.fi/kartat-ja-paikkatieto/koordinaatit-ja-paikannus/koordinaattijarjestelmat

[17] Mäkelä, M. (2024). Konveksi analyysi ja optimointi. Turun yliopisto. Viitattu 15.11.2024. https://intranet.utu.fi/fi/yksikot/sci/yksikot/mattil/opiskelu/kurssit/Documents/moniste_KA_2024.pdf

[18] Mansikkaniemi, H. (1964). Main features of the glacial and postglacial development of Pulmanki Valley in northernmost Finland. Turun yliopisto. (Rep. Kevo Subarctic Sta. 1), 322-337.

[19] Mansikkaniemi, H. (1967). Geomorphological analysis of Pulmanki-Tana valley in Lapland. Turun yliopisto. (Rep. Kevo Subarctic Sta. 4), 7-31.

[20] OpenAI. (2023). ChatGPT (GPT-4o). Saatavilla https://chat.openai.com

[21] Parquer, M. N., Collon, P., & Caumon, G. (2017). Reconstruction of channelized systems through a conditioned reverse migration method. Mathematical Geosciences, 49(8), 965–994. https://doi.org/10.1007/s11004-017-9700-3

[22] Parquer, M., Yan, N., Colombera, L., Mountney, N. P., Collon, P., & Caumon, G. (2020). Combined inverse and forward numerical modelling for reconstruction of channel evolution and facies distributions in fluvial meander-belt deposits. Marine and Petroleum Geology, 117, 104409-. https://doi.org/10.1016/j.marpetgeo.2020.104409

[23] Savitzky, A., & Golay, M. J. E. (1964). Smoothing and differentiation of data by simplified least squares procedures. Analytical Chemistry (Washington), 36(8), 1627–1639. https://doi.org/10.1021/ac60214a047

[24] Siikamäki, P., & Hamunen, P. (2018). Oulankajoki: muutoksen virta. Docendo Oy.

[25] Smith, S. (2013). Digital signal processing: A practical guide for engineers and scientists (3rd ed.). Elsevier Science.

[26] Toponogov, V. A., & Rovenski, V. Y. (2006). Differential geometry of curves and surfaces: A concise guide. Birkhäuser.

Avoimet paikkatietoaineistot

[27] Geologian tutkimuskeskus. (2010). *Maaperä 1:200 (maalajit)*.

[28] Maanmittauslaitos. (2015). *Ortokuva*. Lehdet: X5123F, X5124E.

[29] Maanmittauslaitos. (2022). *Ortokuva*. Lehdet: T5312D, T5312F, T5312H, T5312G, T5314A.

[30] Suomen ympäristökeskus. (2012). *Uomaverkosto 1:10*.

Liite: Lähdekoodi

Liitteessä esitetty lähdekoodi ja tutkielmassa käytetyt aineistot löytyvät osoitteesta `https://github.com/oonaoksanen/migration.git`. Liitteen koodi käyttää Oulankajokea esimerkkinä analyysien ja simulaation suorittamiseen.

Algorithm to simulate the forward migration of meandering river channels.

This code contains an algorithm to analyze and simulate the behavior of meandering river channel. It uses geospatial data and mathematical computations to model the river's future migration and creates a simulation of potential future changes in the river channel.

The code below uses the Oulankajoki River as an example. By modifying the input parameters, such as the centerline, feasible region, and other required variables, this method can also be used to model other meandering rivers.

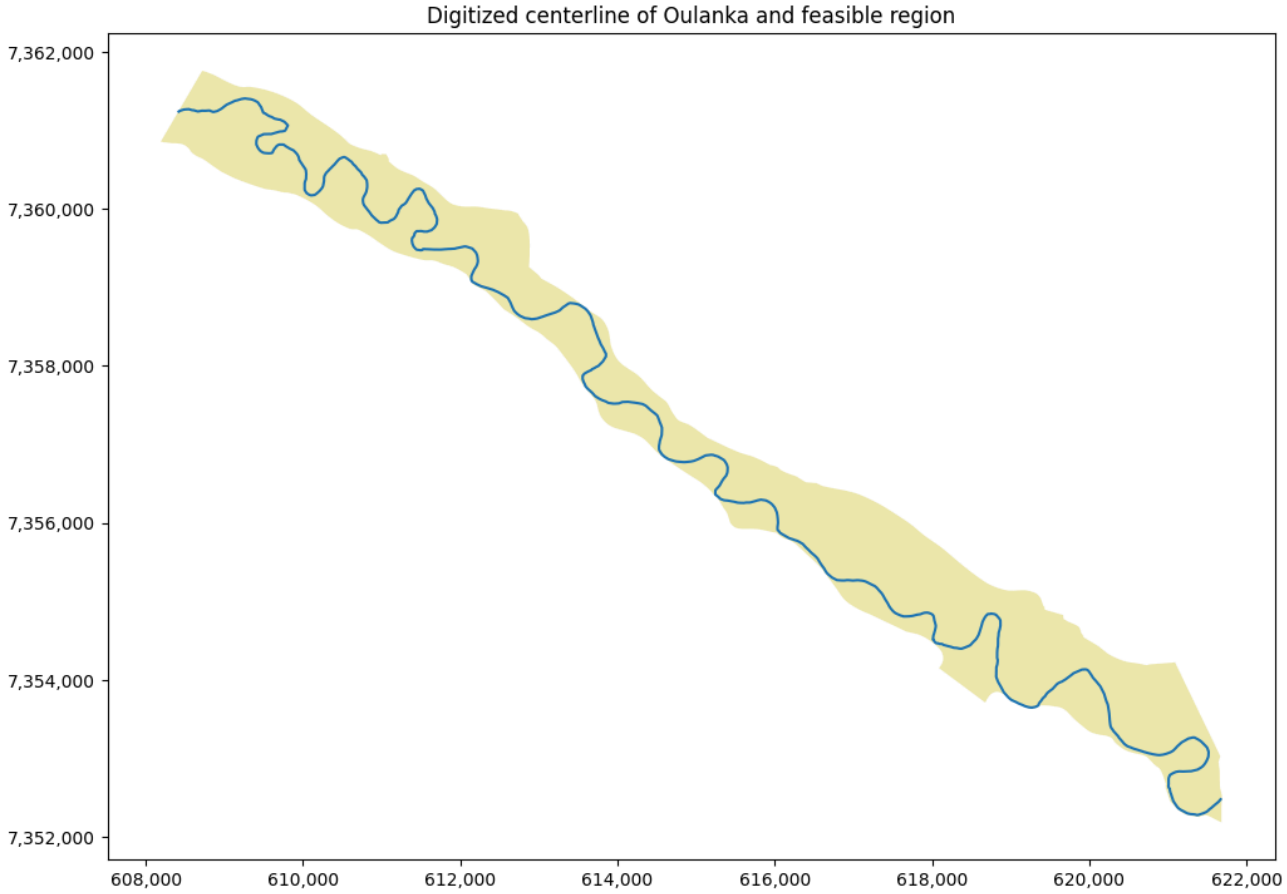
Algorithm is part of Oona Oksanen's master's thesis (Pro gradu).

Load the river channel **centerline** and **feasible region** from shapefiles into GeoDataFrames for further processing.

In [1]:
```
import geopandas as gpd

# River channel centerline
centerline = gpd.read_file(r'C:\data\oulankajoki.shp')

# Feasible region
feasible_region = gpd.read_file(r'C:\data\sallittualue_maalajit_oulanka.shp')
```

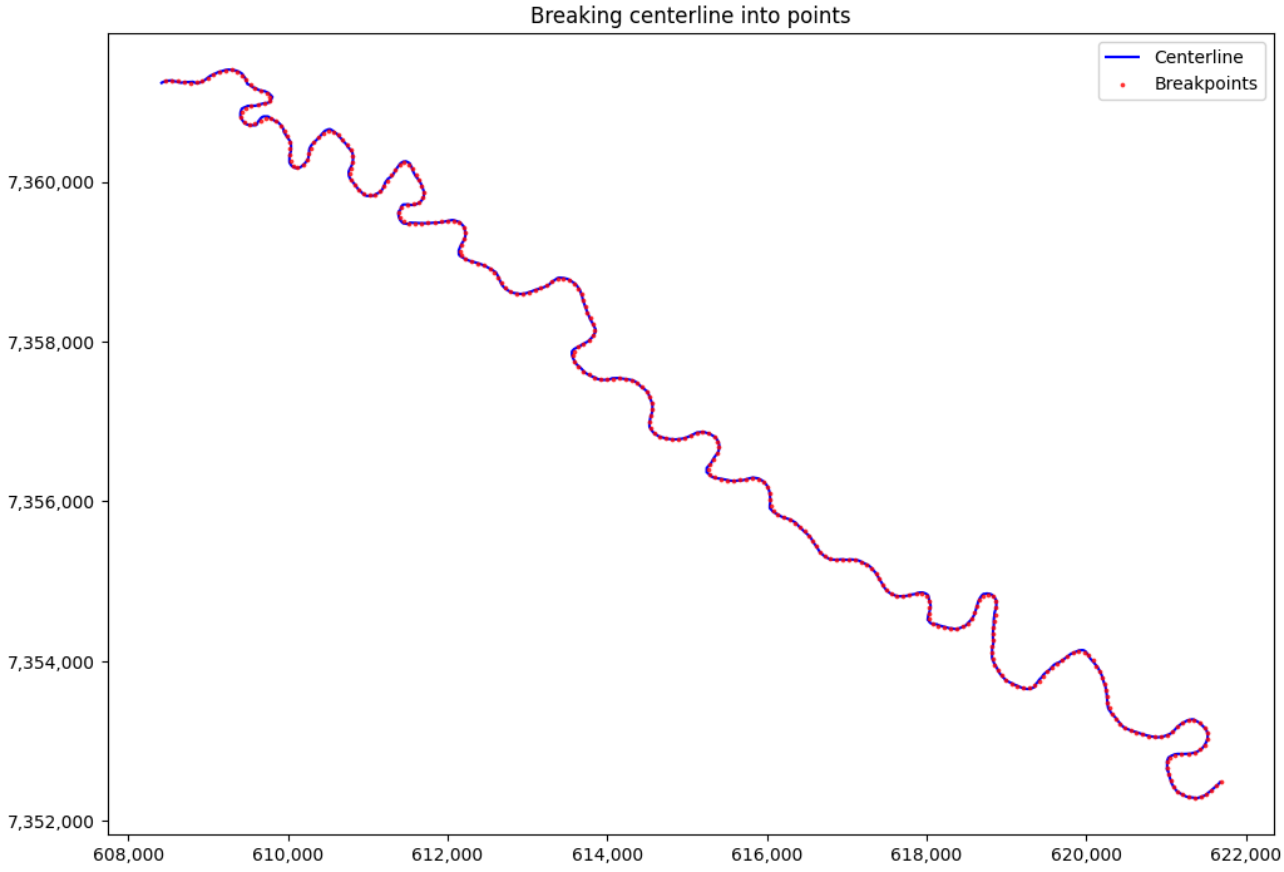
Plot the centerline, feasible region and coordinates.

In [2]:
```
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter

# Plot
fig, ax = plt.subplots(figsize=(12, 9))

def format_eastings(value, pos):
    return f'{int(value):,}'

def format_northings(value, pos):
    return f'{int(value):,}'

ax.xaxis.set_major_formatter(FuncFormatter(format_eastings))
ax.yaxis.set_major_formatter(FuncFormatter(format_northings))

feasible_region.plot(ax=ax, color = 'palegoldenrod')

centerline.plot(ax=ax)

plt.title("Digitized centerline of Oulanka and feasible region")

plt.show()
```



Divide the centerline into **breakpoints** at given interval and smooth them using the Savitzky-Golay filter.

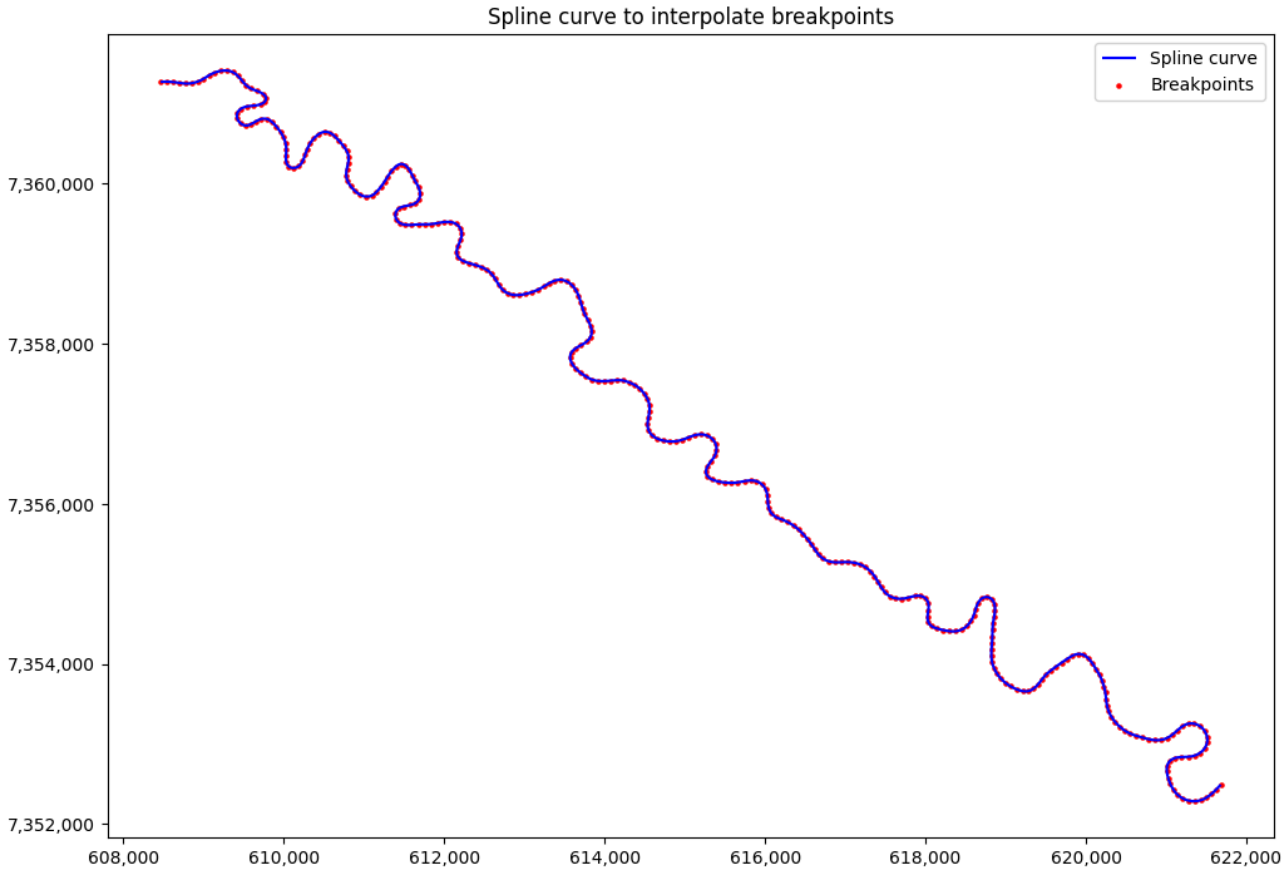
In [3]:
```
from shapely.geometry import LineString, Point
from scipy.signal import savgol_filter

# GeoDataFrame
points = gpd.GeoDataFrame(columns=['geometry'])

# Function to create breakpoints at given interval
def create_breakpoints(geometry, interval):

    breakpoints = LineString(geometry).interpolate(range(0, int(geometry.length), interval))

    #Smooth points using Savitzky-Golay filtering
    x_coords = [point.x for point in breakpoints]
    y_coords = [point.y for point in breakpoints]
    smoothed_x = savgol_filter(x_coords, window_length = 7, polyorder=3)
    smoothed_y = savgol_filter(y_coords, window_length = 7, polyorder=3)

    smoothed_breakpoints = [Point(x, y) for x, y in zip(smoothed_x, smoothed_y)]

    # Add points to gdf
    point = gpd.GeoDataFrame(geometry=smoothed_breakpoints, crs='EPSG:3067')
    return point[::-1]

# Divide points
points = create_breakpoints(centerline['geometry'].iloc[0], 80)

# Plot
fig, ax = plt.subplots(figsize=(12,9))
def format_eastings(value, pos):
    return f'{int(value):,}'

def format_northings(value, pos):
    return f'{int(value):,}'

ax.xaxis.set_major_formatter(FuncFormatter(format_eastings))
ax.yaxis.set_major_formatter(FuncFormatter(format_northings))

centerline.plot(ax=ax, color = 'blue', zorder = 1, linewidth = 1.5, label = 'Centerline')
points.plot(ax=ax, color='red', marker='o', markersize=3, alpha = 0.7, label = 'Breakpoints')

plt.title("Breaking centerline into points")

ax.legend()
plt.show()
```



Interpolate the breakpoints to reconstruct a continuous line.

In [4]:
```
import geopandas as gpd
import numpy as np
from scipy.interpolate import splprep, splev
from shapely.geometry import LineString, Point
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter

def create_spline(points):

    x, y = zip(*points)

    # Spline parameters
    tck, u = splprep([x, y], s=3)
    t_new = np.linspace(0, 1, 1000)
    spline_points = splev(t_new, tck)

    # Convert to Points
    spline_points = [Point(coord) for coord in zip(spline_points[0], spline_points[1])]
    bspline = LineString(spline_points)
    bspline_gdf = gpd.GeoDataFrame(geometry=[bspline])

    return bspline, bspline_gdf

# Generate spline curve
spline_ls, spline_gdf = create_spline(points['geometry'].apply(lambda point: (point.x, point.y)).tolist())

# Plot
fig, ax = plt.subplots(figsize=(12,9))

def format_eastings(value, pos):
    return f'{int(value):,}'

def format_northings(value, pos):
    return f'{int(value):,}'

ax.xaxis.set_major_formatter(FuncFormatter(format_eastings))
ax.yaxis.set_major_formatter(FuncFormatter(format_northings))

spline_gdf.plot(ax=ax, color='blue', label = 'Spline curve')

points.plot(ax=ax, color='red', marker='o', markersize=5, label = 'Breakpoints')

plt.title("Spline curve to interpolate breakpoints")
```

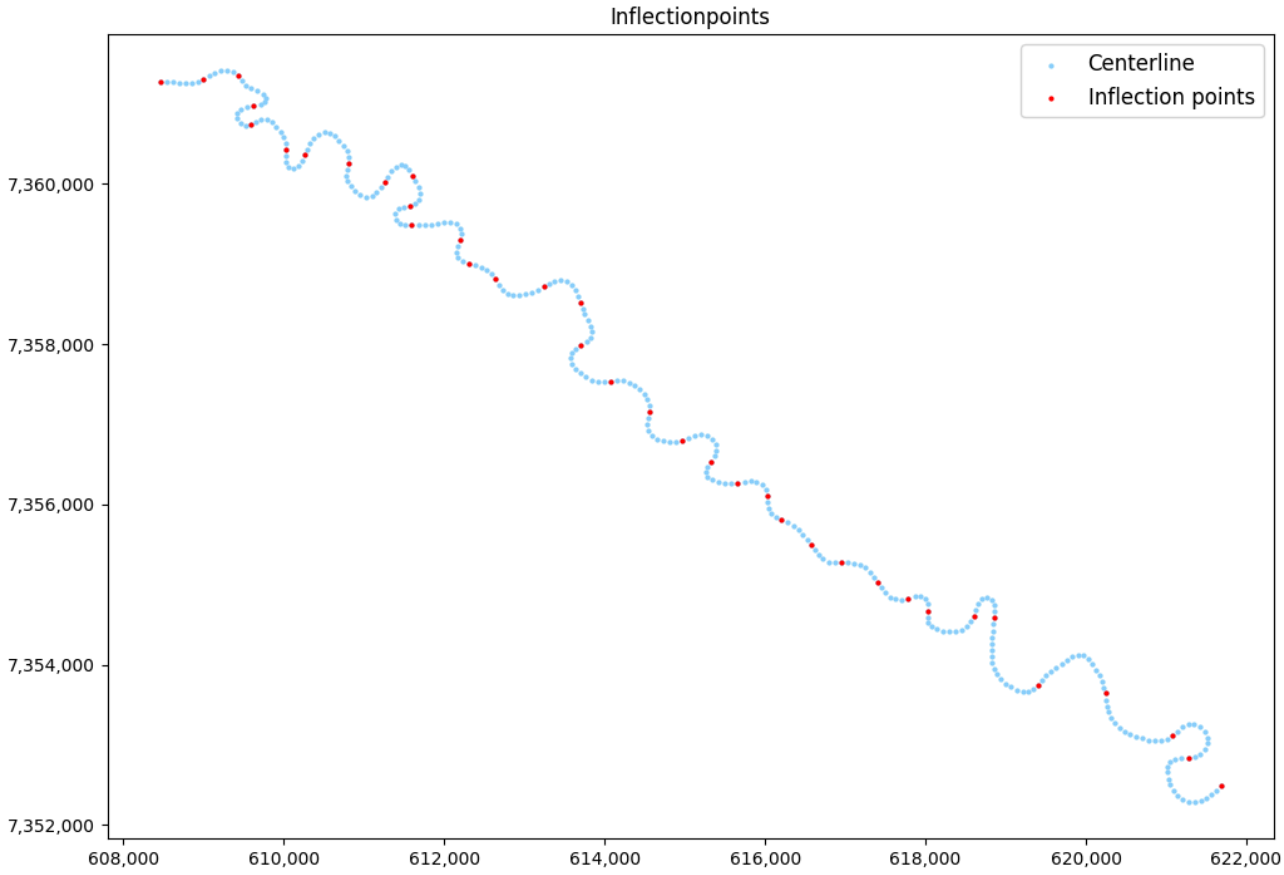
```
ax.legend()
plt.show()
```



Detect **inflection points** by calculating dot products of consecutive vectors $u_i \cdot \hat{v}_i$ and observe sign changes.

In [5]:
```
#Inflectionpoints defined by u_i * v_i^

import numpy as np

def define_inflectionpoints(points, p_between):
    inflectionpoint_list = []
    points_in_between = 0
    inflectionpoint_list.append(points.iloc[0]) # First point

    for i in range(1, len(points)-2):

        # Vectors
        v1 = (np.array(points.iloc[i-1].coords[0])) - (np.array(points.iloc[i].coords[0]))

        v2 = (np.array(points.iloc[i].coords[0])) - (np.array(points.iloc[i+1].coords[0]))
        v2_T = np.array([-v2[1], v2[0]])

        v3 = (np.array(points.iloc[i+1].coords[0])) - (np.array(points.iloc[i+2].coords[0]))
        v3_T = np.array([-v3[1], v3[0]])

        # Dot product
        dot_product1 = np.dot(v1, v2_T) # u_i * v_i^
        dot_product2 = np.dot(v2, v3_T) # u_i+1 * v_i+1^

        # Counter to ensure that there are at least 4 points between inflection points
        points_in_between = points_in_between + 1

        # Check when sign changes
        if np.sign(dot_product1) != np.sign(dot_product2) and points_in_between > p_between:
            inflectionpoint_list.append(points.iloc[i]) # Add to inflection point list

            points_in_between = 0
    inflectionpoint_list.append(points.iloc[len(points)-1]) # Last point

    return inflectionpoint_list

# Define inflection points
inflection_points = define_inflectionpoints(points['geometry'], 4)

# Plot
fig, ax = plt.subplots(figsize=(12, 9))

def format_eastings(value, pos):
```

```
    return f'{int(value):,}'

def format_northings(value, pos):
    return f'{int(value):,}'

ax.xaxis.set_major_formatter(FuncFormatter(format_eastings))
ax.yaxis.set_major_formatter(FuncFormatter(format_northings))

points.plot(ax=ax, color='lightskyblue', marker='o', markersize=4, label='Centerline')
gpd.GeoDataFrame(geometry=inflection_points).plot(ax=ax, color='red', marker='o', markersize=4,
                                                  label='Inflection points')

plt.title("Inflectionpoints")

ax.legend(fontsize = 12)
plt.show()
```



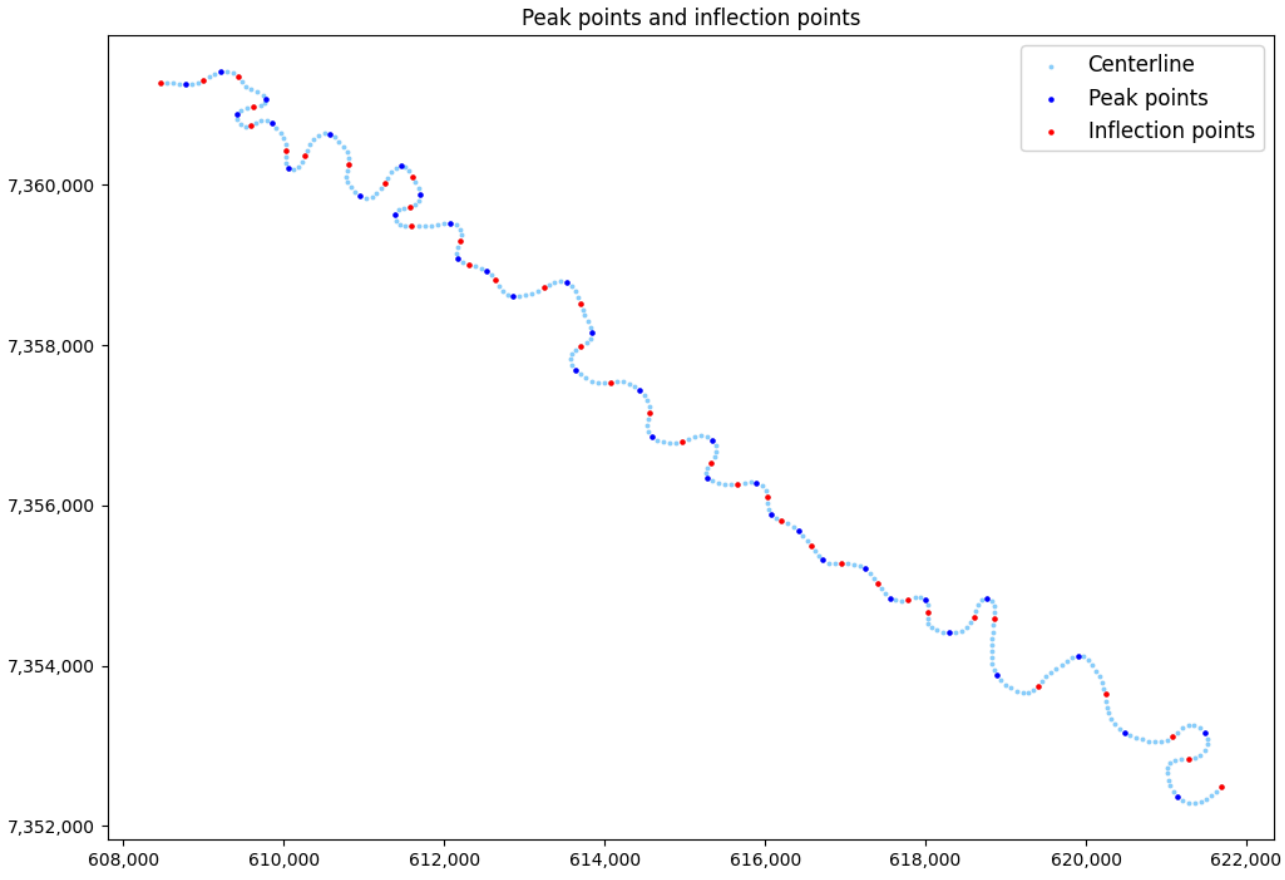
Define **meander bends** in the GeoDataFrame based on previously defined inflection points.

In [6]:
```
# Define meander bends

def define_meander(points, inflectionpoints):
    points['meander'] = None
    meander_id = 0
    j = 1

    for i in range(len(points['geometry'])):
        if points['geometry'].iloc[i].coords[0] == inflectionpoints[j].coords[0]:
            points.at[len(points) - i - 1, 'meander'] = meander_id
            meander_id += 1
            if j != len(inflectionpoints) - 1:
                j += 1
        else:
            points.at[len(points) - i - 1, 'meander'] = meander_id

define_meander(points, inflection_points)
```

Define the **peak points** of meander bends as the points with the maximum distance from the midpoint of the line between two inflection points.

In [7]:
```
from shapely.geometry import Point, LineString

def define_peak_points(inflection_points, points):
    farthest_points = []
    max_distance = 0
```

```
    for i in range(len(inflection_points) - 1):
        p1 = inflection_points[i]
        p2 = inflection_points[i + 1]

        line = LineString([p1, p2])

        meander_points = points[points['meander'] == i]

        dist = []

        # Calculate distances of all points from line
        for j in range(len(meander_points)):
            distance = line.distance(meander_points['geometry'].iloc[j])
            dist.append(distance)

        max_distance = max(dist)  # Find max distance point
        farthest_point_index = dist.index(max_distance)
        farthest_point = meander_points['geometry'].iloc[farthest_point_index]

        farthest_points.append(farthest_point)

    return farthest_points

peak = define_peak_points(inflection_points, points)

# Plot
fig, ax = plt.subplots(figsize=(12, 9))

def format_eastings(value, pos):
    return f'{int(value):,}'

def format_northings(value, pos):
    return f'{int(value):,}'

ax.xaxis.set_major_formatter(FuncFormatter(format_eastings))
ax.yaxis.set_major_formatter(FuncFormatter(format_northings))

points.plot(ax=ax, color='lightskyblue', marker='o', markersize=3, label= 'Centerline')

gpd.GeoDataFrame(geometry=peak).plot(ax=ax, color='blue', marker='o', markersize=5, label='Peak points')
gpd.GeoDataFrame(geometry=inflection_points).plot(ax=ax, color='red', marker='o', markersize=5,
                                                  label='Inflection points')

plt.title("Peak points and inflection points")

ax.legend(fontsize = 12)
plt.show()
```



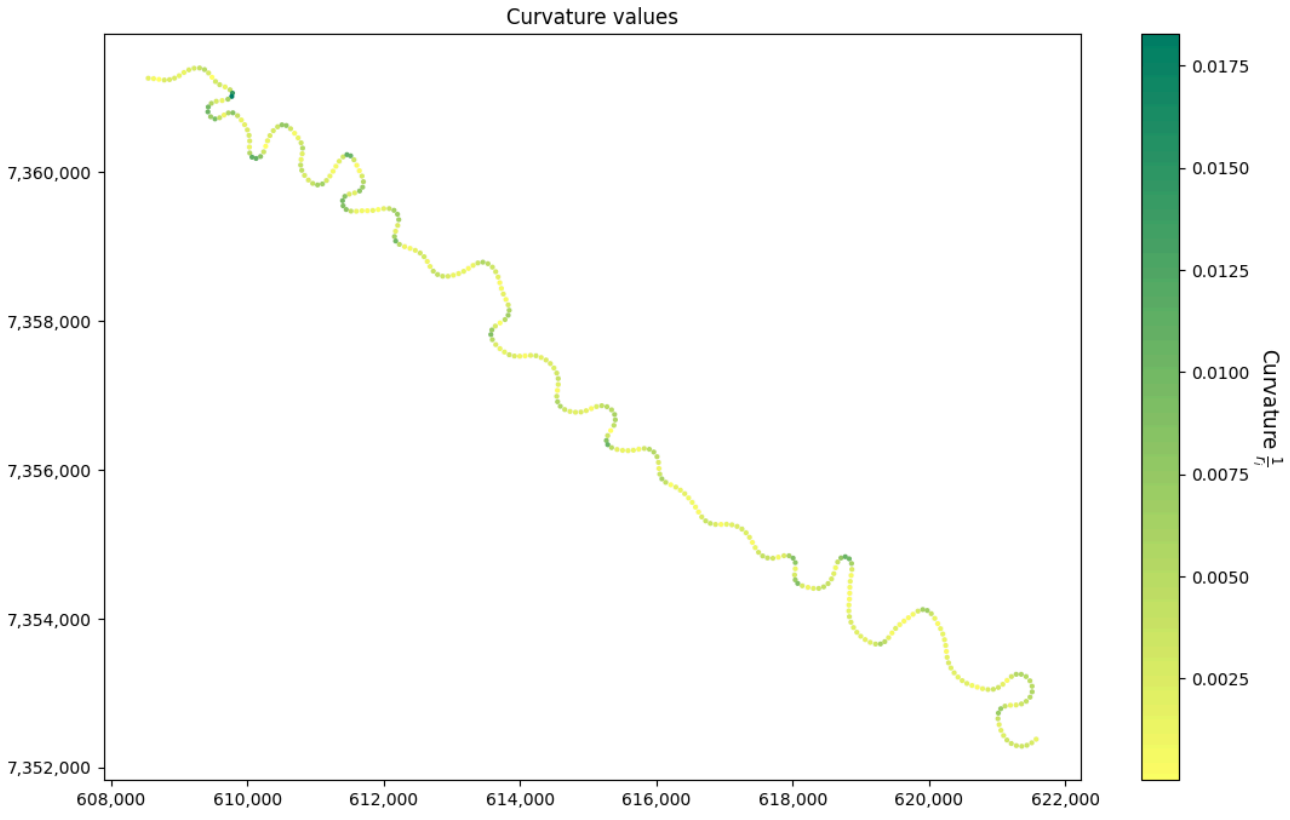
Calculate the radius of circle that intersects three consecutive points.

In [8]:
```
import numpy as np

def radius_and_intersectionpoint(points):
    radius_list = []
    intersectionpoint_list = []

    for i in range(len(points) - 2):

        # Three consecutive points
        p1 = np.array(points['geometry'].iloc[i].coords[0])
        p2 = np.array(points['geometry'].iloc[i + 1].coords[0])
        p3 = np.array(points['geometry'].iloc[i + 2].coords[0])

        midpoint1 = 0.5 * (p1 + p2)
        normal1 = np.array([-(p2 - p1)[1], (p2 - p1)[0]])

        midpoint2 = 0.5 * (p2 + p3)
        normal2 = np.array([-(p3 - p2)[1], (p3 - p2)[0]])

        A = np.array([[normal1[0], -normal2[0]], [normal1[1], -normal2[1]]])
        B = np.array([midpoint2[0] - midpoint1[0], midpoint2[1] - midpoint1[1]])

        if np.linalg.det(A) == 0:
            print("Singular matrix. Skipping this point.")
            continue
        else:
            solution = np.linalg.solve(A, B)

            # intersection
            intersection_points = midpoint1 + solution[1] * normal1
            intersectionpoint_list.append(intersection_points)

            # radius
            radius = np.linalg.norm(intersection_points - p2)
            radius_list.append(radius)

    return radius_list, intersectionpoint_list, solution, A, B

rad, inter, sol, A, B = radius_and_intersectionpoint(points)
```

Calculate the **curvature** $\kappa(x_i) = \frac{1}{r_i}$ for each point.

In [9]:
```
#Curvature 1/r

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import FuncFormatter

def calculate_curvature(points, radius):
    curvature_list = []
    shifted_radius = [np.nan] + radius[:-1]

    points['curvature'] = np.nan

    for i in range(len(points) - 2):
        radius = shifted_radius[i]
        curvature = 1 / radius
        curvature_list.append(curvature)

    points.loc[points.index[:-2], 'curvature'] = curvature_list


calculate_curvature(points, rad)

# Plot
fig, ax = plt.subplots(figsize=(13, 8))
scatter = ax.scatter(
    points.geometry.x,
    points.geometry.y,
    c=points['curvature'],
    cmap= 'summer_r',
    s=4
)

cbar = plt.colorbar(scatter, label='Curvature')
cbar.set_label(r'Curvature $\frac{1}{r_i}$', rotation=270, labelpad=20, fontsize = 12)

def format_eastings(value, pos):
    return f'{int(value):,}'

def format_northings(value, pos):
    return f'{int(value):,}'
```

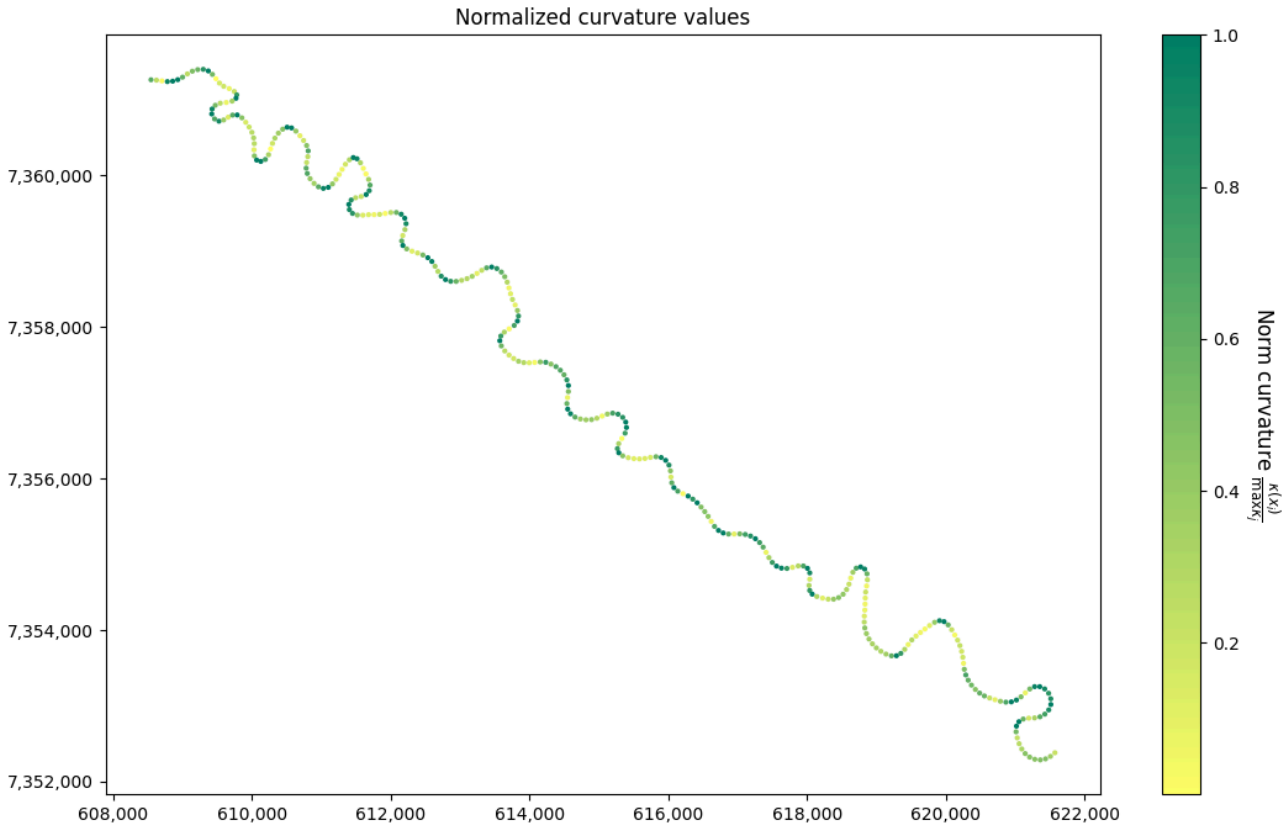
```
ax.xaxis.set_major_formatter(FuncFormatter(format_eastings))
ax.yaxis.set_major_formatter(FuncFormatter(format_northings))

plt.title("Curvature values")

plt.show()
```



Normalize curvature values within each meander by dividing the curvature at each point by the maximum curvature of the meander $\frac{\kappa(x_i)}{\max \kappa_j}$.

In [10]:
```
# Norm curvature

def define_normalized_curvature(points):
    points['norm_curvature'] = None

    for i in range(max(points['meander'])+1):

            pm = (points['meander'] == i)
            filtered_pm = points[pm]

            maxcurvature = filtered_pm['curvature'].max()

            for index in filtered_pm.index:
                points.loc[index, 'norm_curvature'] = filtered_pm.loc[index, 'curvature'] / maxcurvature

define_normalized_curvature(points)

# Plot

fig, ax = plt.subplots(figsize=(13, 8))
scatter = ax.scatter(
    points.geometry.x,
    points.geometry.y,
    c=points['norm_curvature'],
    cmap='summer_r',
    s=4
)

cbar = plt.colorbar(scatter, label='Norm curvature')
cbar.set_label(r'Norm curvature $\frac{\kappa(x_i)}{\max \kappa_j}$', rotation=270, labelpad=25, fontsize = 12)

def format_eastings(value, pos):
    return f'{int(value):,}'

def format_northings(value, pos):
    return f'{int(value):,}'

ax.xaxis.set_major_formatter(FuncFormatter(format_eastings))
ax.yaxis.set_major_formatter(FuncFormatter(format_northings))
```

```
plt.title("Normalized curvature values")

plt.show()
```



Determine if the point is contained within the **feasible region**.

In [11]:
```
def check_feasible_region(prev, point, feasibleregion):

    if Point(point).within(feasibleregion.geometry.iloc[0]):
        new_p = Point(point)
    else:
        new_p = prev

    return new_p
```

Formation of **oxbow lakes**.

In [12]:
```
import geopandas as gpd
import pandas as pd

# Oxbow Lakes

def new_oxbow(points, inflection, peak, meters):
    new_points = points.copy()
    oxbows = []

    # Check distances between inflection points and peak points
    for i in range(len(peak)):
        for j in range(i + 1, len(peak)):

            if inflection[i].distance(inflection[j]) < meters \
            or peak[i].distance(peak[j]) < meters \
            or inflection[i].distance(peak[j]) < meters:

                min_meander = min(i, j)
                max_meander = max(i, j)

                # Select all points that lie between the points to be connected
                oxbow_points = new_points[(new_points['meander'] >= min_meander) & (new_points['meander'] <= max_meander)]

                if not oxbow_points.empty:
                    oxbows.append(gpd.GeoDataFrame(oxbow_points, geometry='geometry'))

                # Update river channel points without oxbow points
                new_points = new_points[~((new_points['meander'] >= min_meander) & (new_points['meander'] <= max_meander))]
```

```
    if oxbows:
        oxbow_gdf = pd.concat(oxbows, ignore_index=True)
    else:
        oxbow_gdf = gpd.GeoDataFrame()

    return new_points, oxbow_gdf
```

Define the **directions D** and **L**, where **D** is the direction between inflection points, and **L** is the direction toward the peak point.

In [13]:
```
import math

def define_directions_peak(inflectionpoints, peakpoints):
    D_list = []
    L_peak_list = []
    j = 0

    for i in range(len(inflectionpoints) - 1):

        # D points
        x1_inf, y1_inf = inflectionpoints[i].x, inflectionpoints[i].y
        x2_inf, y2_inf = inflectionpoints[i + 1].x, inflectionpoints[i + 1].y

        # Direction of D
        v_inf = (x2_inf - x1_inf, y2_inf - y1_inf)

        # Length of D
        v_inf_length = math.sqrt(v_inf[0] ** 2 + v_inf[1] ** 2)

        # Norm D
        D = (v_inf[0] / v_inf_length, v_inf[1] / v_inf_length)

        # L points
        x_mid , y_mid  = 0.5*(x1_inf + x2_inf) , 0.5*(y1_inf + y2_inf)
        x_peak, y_peak = peakpoints[j].x, peakpoints[j].y
        j = j+1

        # Direction L
        v_peak = (x_peak - x_mid, y_peak - y_mid)

        # Length L
        v_peak_length = math.sqrt(v_peak[0] ** 2 + v_peak[1] ** 2)

        # Norm L
        L = (v_peak[0] / v_peak_length, v_peak[1] / v_peak_length)

        D_list.append(D)
        L_peak_list.append(L)

    return D_list, L_peak_list

D, L = define_directions_peak(inflection_points, peak)
```

Define **migration offsets** $O_D(x_i)$ and $O_L(x_i)$.

Model is created by Parquer, M. N., Collon, P., & Caumon, G. (2017). Reconstruction of Channelized Systems Through a Conditioned Reverse Migration Method.

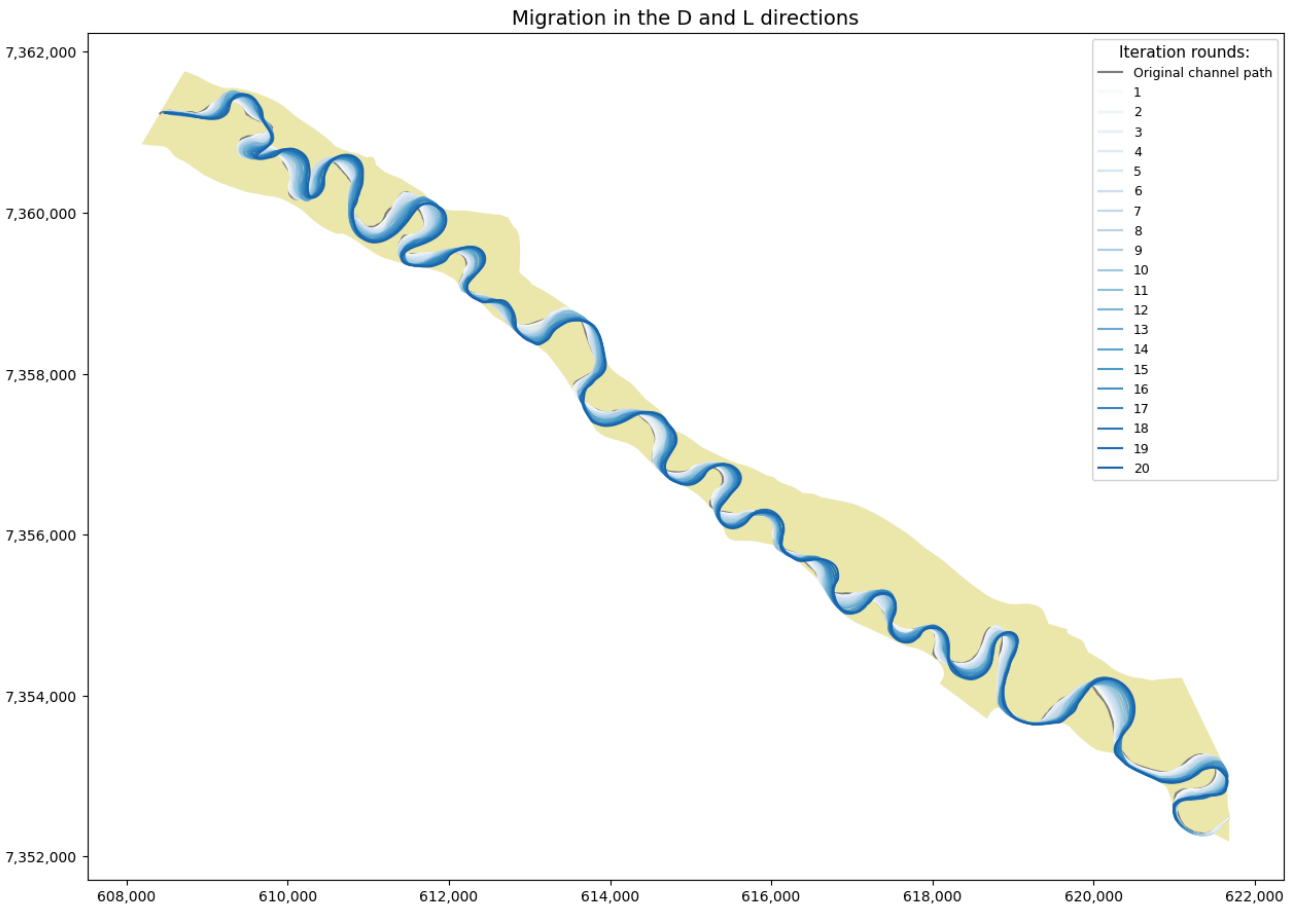
In [14]:
```
def offsets(points, OD, OL, sD, sL, w):
    od_xi = []
    ol_xi = []

    max_curvature=points['curvature'].max()

    for i in range(len(points)):

        if w > 0:
            OD_xi = OD - (sD * (points['curvature'].iloc[i] / max_curvature))

            OL_xi = OL - (sL * (1 - (points['curvature'].iloc[i] / max_curvature)))

        elif w < 0:
            OD_xi = OD - (sD * (1 - (points['curvature'].iloc[i] / max_curvature)))

            OL_xi = OL - (sL * (points['curvature'].iloc[i] / max_curvature))


        od_xi.append(OD_xi)
        ol_xi.append(OL_xi)

    return od_xi, ol_xi
```

Algorithm for modeling future migration of a meandering river channel.

In [15]:
```
#ALGORITHM

import numpy as np
from shapely.geometry import Point

def reverse_migration_simulation(N_s, channel_path, mean_offset, std_dev_offset, cut_off_distance, feasible_region):
    new_points = {}
    spline = {}
    spline_gdf = {}
    oxbow_points = []

    for s in range(N_s):

        # Initialize a GeoDataFrame to store the new points
        new_points[s] = gpd.GeoDataFrame(columns=['geometry', 'curvature', 'meander'])

        # Inflection points of the current channel path
        inflection_points = define_inflectionpoints(channel_path['geometry'], 4)

        # Define meander bends
        define_meander(channel_path, inflection_points)
        new_points[s]['meander'] = channel_path['meander']

        # Define peak points
        peak = define_peak_points(inflection_points, channel_path)

        # Radius of circles which are defined by three consecutive points
        radius, interp, sol, A, B = radius_and_intersectionpoint(channel_path)

        # Curvature
        calculate_curvature(channel_path, radius)
        new_points[s]['curvature'] = channel_path['curvature']

        # Define the Nhm number of half-meanders of current channel path
        N_hm = len(inflection_points) - 1

        # Define the migration directions
        D_direction, L_direction = define_directions_peak(inflection_points, peak)

        # Sample horizontal and lateral offsets from Gaussian distribution

        '''

        By changing these:

        1.Pure lateral migration : OD = 0 and OL ~ N(mean,sd),

        2.Pure downstream migration : OD ~ N(mean, sd) and  OL = 0,

        3.Mixed migration : OD ~ N(mean, sd) and OL ~ N(mean,sd),

        '''

        OD = np.random.normal(mean_offset, std_dev_offset, N_hm)
        #OD = np.zeros(N_hm)

        OL = np.random.normal(mean_offset, std_dev_offset, N_hm)
        #OL = np.zeros(N_hm)

        # Sample smoothing values sL and sD from a uniform distribution

        min_sD = 0
        max_sD = 2*OD

        min_sL = 0
        max_sL = 2*OL

        sD = np.random.uniform(min_sD, max_sD)

        sL = np.random.uniform(min_sL, max_sL)

        # Sample the weighting w
        w = np.random.choice([-1, 1], N_hm)

        # Initialize a list to store the new points
        newpoint_list = []

        for hm in range(N_hm):

            # Filter points of one meander
            pm = (channel_path['meander'] == hm)
            filtered_pm = channel_path[pm]

            mig_D, mig_L = offsets(filtered_pm, OD[hm], OL[hm], sD[hm], sL[hm], w[hm])
```

```
            # Move points

            for pn in range(len(filtered_pm['geometry'])):

                new_point = (
                    filtered_pm['geometry'].iloc[pn].x + ((mig_D[pn] * D_direction[hm][0])
                                                          + (mig_L[pn] *  L_direction[hm][0])),
                    filtered_pm['geometry'].iloc[pn].y + ((mig_D[pn] * D_direction[hm][1])
                                                          + (mig_L[pn] *  L_direction[hm][1]))
                    )
                if not np.isnan(new_point).all():
                    newpoint = check_feasible_region(filtered_pm['geometry'].iloc[pn], new_point, feasible_region)
                    newpoint_list.append(newpoint)
                else:
                    newpoint = (filtered_pm['geometry'].iloc[pn].x ,  filtered_pm['geometry'].iloc[pn].y)
                    newpoint_list.append(Point(newpoint))


        # Add new points to the GeoDataFrame
        new_points[s]['geometry'] = newpoint_list

        # Check if new oxbow lakes will appear
        channel_path_deleted_oxbows, oxbows = new_oxbow(channel_path, inflection_points, peak, cut_off_distance)


        # If there is new oxbow lakes
        if len(oxbows) > 0:

            # Add points without oxbow bends
            new_points[s]=gpd.GeoDataFrame(columns=['geometry'])
            new_points[s]['geometry'] = [Point((point.x, point.y)) for point in channel_path_deleted_oxbows['geometry']]

            oxbow_points.append(oxbows)

        # Spline
        spline[s], spline_gdf[s] = create_spline((new_points[s]['geometry']).apply(lambda point: (point.x, point.y)).tolist())

        # Update new path to next round
        channel_path = create_breakpoints(spline[s], 80)
        channel_path['geometry'] = channel_path['geometry'].iloc[::-1].values

    # Plot

    fig, ax = plt.subplots(figsize =(15,12))

    def format_eastings(value, pos):
        return f'{int(value):,}'

    def format_northings(value, pos):
            return f'{int(value):,}'

    ax.xaxis.set_major_formatter(FuncFormatter(format_eastings))
    ax.yaxis.set_major_formatter(FuncFormatter(format_northings))

    feasible_region.plot(ax=ax, color = 'palegoldenrod')

    # Original channel_path
    centerline.plot(ax=ax, color='grey', label='Original channel path')

    # Splines of rounds
    for i in range (len(new_points)):
        gradient = np.linspace(0, 1, len(new_points)+5)
        color = plt.cm.Blues(gradient[i])
        gpd.GeoDataFrame(geometry=[spline[i]]).plot(ax=ax, color=color, label=f'{i+1}')

    #title_txt = "Migration in the L-direction"
    #title_txt = "Migration in the D-direction"
    title_txt = "Migration in the D and L directions"



    # Set title
    plt.title(title_txt, fontsize=14)
    suptitle_text = f'N({mean_offset},{std_dev_offset})'
    plt.suptitle(suptitle_text, x=0.5, y=0.05, ha='center', fontsize = 14)

    plt.legend(title="Iteration rounds:", title_fontsize=11,  fontsize=9)

    plt.show()


'''

Input:

N_s: Number of simulations.
```

```
channel_path: Current river channel.

mean_offset: Average D and L migration distance.
std_dev_offset: Standard deviation of D and L migration distances.

cut_off: Cut-off distance for oxbow lakes.
feasible_region: Feasible region for channel migration.

'''

N_s = 20

channel_path = points

mean_offset = 20
std_dev_offset = 5

# Negative offset
# mean_offset = 0
# std_dev_offset = 10

cut_off = 100
feasible_region = feasible_region

reverse_migration_simulation(N_s, channel_path, mean_offset, std_dev_offset,  cut_off, feasible_region)
```



N(20,5)

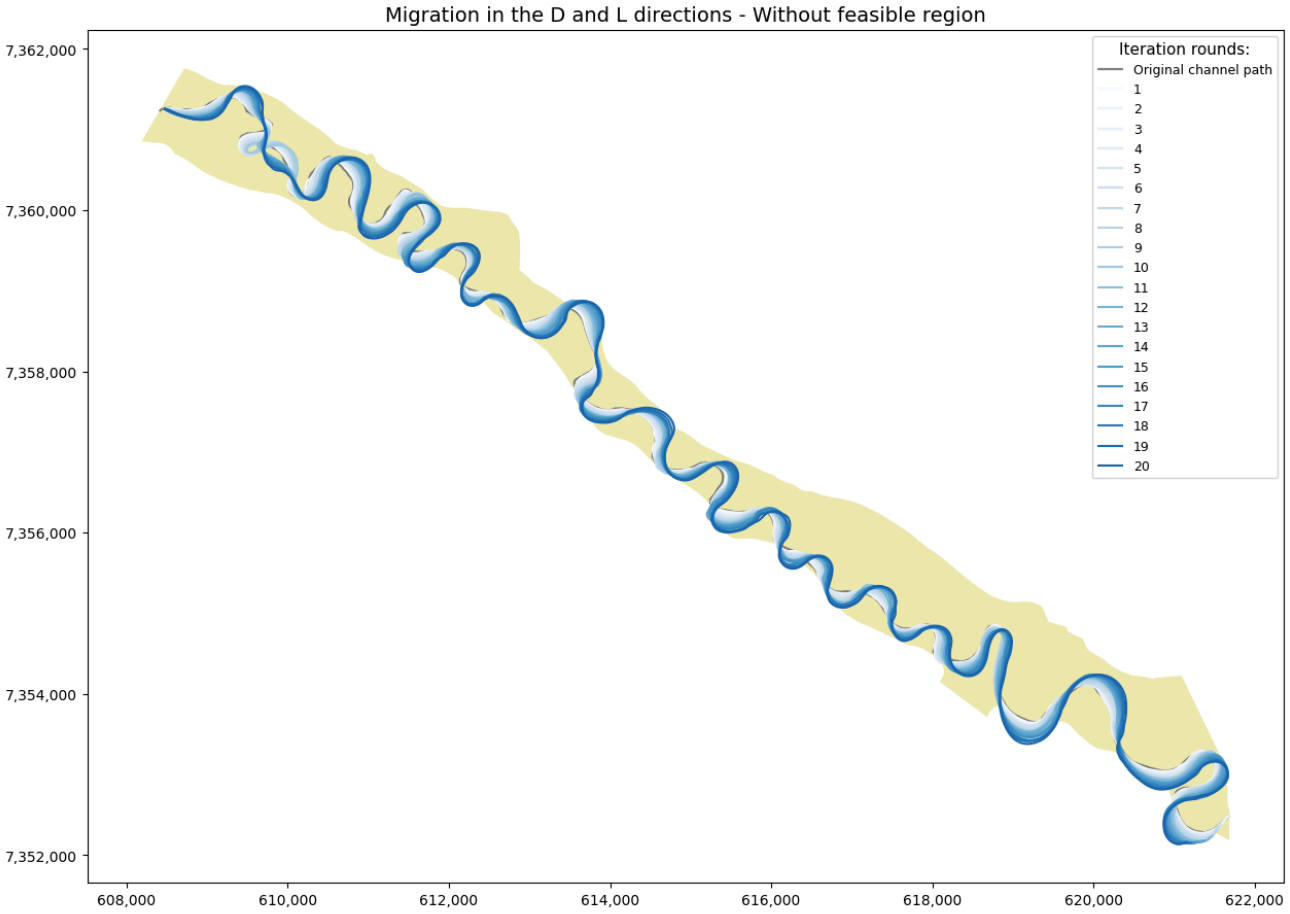
The same algorithm, but **without** using the **feasible region** to constrain migration.

In [16]:
```
#ALGORITHM without feasible region

import numpy as np
from shapely.geometry import Point

def reverse_migration_simulation(N_s, channel_path, mean_offset, std_dev_offset, cut_off_distance, feasible_region):
    new_points = {}
    spline = {}
    spline_gdf = {}
    oxbow_points = []

    for s in range(N_s):
```

```
# Initialize a GeoDataFrame to store the new points
new_points[s] = gpd.GeoDataFrame(columns=['geometry', 'curvature', 'meander'])

# Inflection points of the current channel path
inflection_points = define_inflectionpoints(channel_path['geometry'], 4)

# Define meander bends
define_meander(channel_path, inflection_points)
new_points[s]['meander'] = channel_path['meander']

# Define peak points
peak = define_peak_points(inflection_points, channel_path)

# Radius of circles which are defined by three consecutive points
radius, interp, sol, A, B = radius_and_intersectionpoint(channel_path)

# Curvature
calculate_curvature(channel_path, radius)
new_points[s]['curvature'] = channel_path['curvature']

# Define the Nhm number of half-meanders of current channel path
N_hm = len(inflection_points) - 1

# Define the migration directions
D_direction, L_direction = define_directions_peak(inflection_points, peak)

# Sample horizontal and lateral offsets from Gaussian distribution

'''

By changing these:

1.Pure lateral migration : OD = 0 and OL ~ N(mean,sd),

2.Pure downstream migration : OD ~ N(mean, sd) and  OL = 0,

3.Mixed migration : OD ~ N(mean, sd) and OL ~ N(mean,sd),

'''

OD = np.random.normal(mean_offset, std_dev_offset, N_hm)
#OD = np.zeros(N_hm)

OL = np.random.normal(mean_offset, std_dev_offset, N_hm)
#OL = np.zeros(N_hm)

# Sample smoothing values sL and sD from a uniform distribution

min_sD = 0
max_sD = 2*OD

min_sL = 0
max_sL = 2*OL

sD = np.random.uniform(min_sD, max_sD)

sL = np.random.uniform(min_sL, max_sL)

# Sample the weighting w
w = np.random.choice([-1, 1], N_hm)

# Initialize a list to store the new points
newpoint_list = []

for hm in range(N_hm):

    # Filter points of one meander
    pm = (channel_path['meander'] == hm)
    filtered_pm = channel_path[pm]

    mig_D, mig_L = offsets(filtered_pm, OD[hm], OL[hm], sD[hm], sL[hm], w[hm])

    # Move points, no checking feasible region

    for pn in range(len(filtered_pm['geometry'])):

        new_point = (
            filtered_pm['geometry'].iloc[pn].x + ((mig_D[pn] * D_direction[hm][0])
                                                  + (mig_L[pn] *  L_direction[hm][0])) ,
            filtered_pm['geometry'].iloc[pn].y + ((mig_D[pn] * D_direction[hm][1])
                                                  + (mig_L[pn] *  L_direction[hm][1]))
            )
        if not np.isnan(new_point).all():
            newpoint = Point(new_point)
            newpoint_list.append(newpoint)
        else:
```

```
                    newpoint = Point(filtered_pm['geometry'].iloc[pn].x, filtered_pm['geometry'].iloc[pn].y)
                    newpoint_list.append(newpoint)


        # Add new points to the GeoDataFrame
        new_points[s]['geometry'] = newpoint_list

        # Check if new oxbow lakes will appear
        channel_path_deleted_oxbows, oxbows = new_oxbow(channel_path, inflection_points, peak, cut_off_distance)


        # If there is new oxbow lakes
        if len(oxbows) > 0:

            # Add points without oxbow bends
            new_points[s]=gpd.GeoDataFrame(columns=['geometry'])
            new_points[s]['geometry'] = [Point((point.x, point.y)) for point in channel_path_deleted_oxbows['geometry']]

            oxbow_points.append(oxbows)

        # Spline
        spline[s], spline_gdf[s] = create_spline((new_points[s]['geometry']).apply(lambda point: (point.x, point.y)).tolist())

        # Update new path to next round
        channel_path = create_breakpoints(spline[s], 80)
        channel_path['geometry'] = channel_path['geometry'].iloc[::-1].values

    # Plot

    fig, ax = plt.subplots(figsize =(15,12))

    def format_eastings(value, pos):
        return f'{int(value):,}'

    def format_northings(value, pos):
            return f'{int(value):,}'

    ax.xaxis.set_major_formatter(FuncFormatter(format_eastings))
    ax.yaxis.set_major_formatter(FuncFormatter(format_northings))

    feasible_region.plot(ax=ax, color = 'palegoldenrod')

    # Original channel_path
    centerline.plot(ax=ax, color='grey', label='Original channel path')

    # Splines of rounds
    for i in range (len(new_points)):
        gradient = np.linspace(0, 1, len(new_points)+5)
        color = plt.cm.Blues(gradient[i])
        gpd.GeoDataFrame(geometry=[spline[i]]).plot(ax=ax, color=color, label=f'{i+1}')

    #title_txt = "Migration in the L-direction - Without feasible region"
    #title_txt = "Migration in the D-direction - Without feasible region"
    title_txt = "Migration in the D and L directions - Without feasible region"



    # Set title
    plt.title(title_txt, fontsize=14)
    suptitle_text = f'N({mean_offset},{std_dev_offset})'
    plt.suptitle(suptitle_text, x=0.5, y=0.05, ha='center', fontsize = 14)

    plt.legend(title="Iteration rounds:", title_fontsize=11,  fontsize=9)

    plt.show()


'''

Input:

N_s: Number of simulations.

channel_path: Current river channel.

mean_offset: Average D and L migration distance.
std_dev_offset: Standard deviation of D and L migration distances.

cut_off: Cut-off distance for oxbow lakes.
feasible_region: Feasible region for channel migration.

'''

N_s = 20

channel_path = points
```

```
mean_offset = 20
std_dev_offset = 5

# Negative offset
#mean_offset = 0
#std_dev_offset = 10

cut_off = 100 #oulanka

feasible_region = feasible_region

reverse_migration_simulation(N_s, channel_path, mean_offset, std_dev_offset,  cut_off, feasible_region)
```



N(20,5)