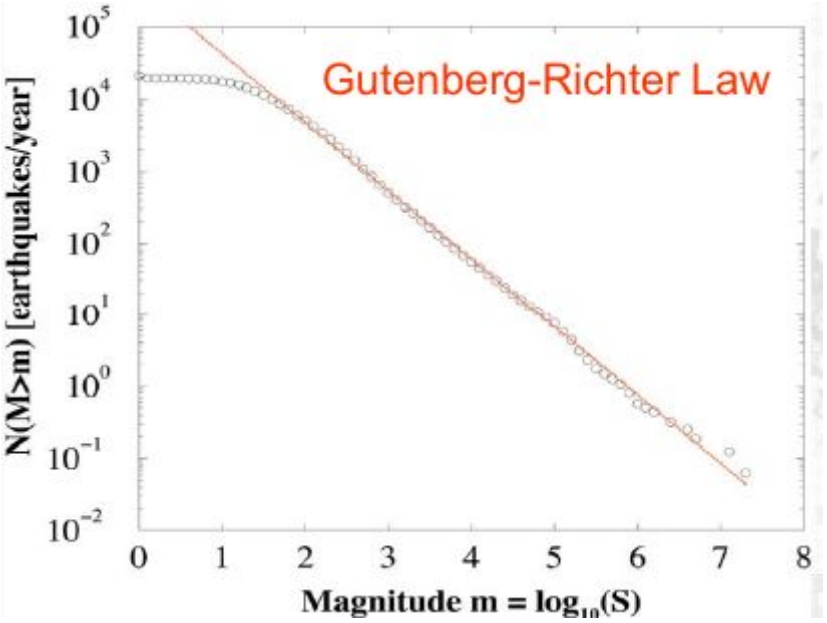
Afsluttende projekt i Datalogi for Fysikere Januar 2018

SOC model af jordskælv

Introduktion:

Jordskælv forekommer mange steder på jorden. Nemlig de steder hvor de tektoniske plader støder sammen. Jordskælv er resultatet en process hvor det opbygget stres mellem to tektoniske plader frigives og resultere i en lokal relativ forskydning af pladerne. Jordskælv findes i mange styrker og tegner man fordelingsfunktionen for antallet af skælv med en bestemt energifrigørelse følger den potens lov som det fremgår af dette billede - den såkaldte Guttenberg-Richter lov.



Figuren viser at der er et meget stort interval i frigivet energi (S) over hvilken der ikke er nogle karakteristisk forskel mellem begivenheder af forskellig størrelse, ud over deres hyppighed. Det tolkes oftest som at der ikke er en karakteristisk længdeskala for fænomenet (som vi kan observere med vores instrumenter) og at begivenheder (her jordskælv) af forskellig størrelse derfor kan beskrives på samme måde.

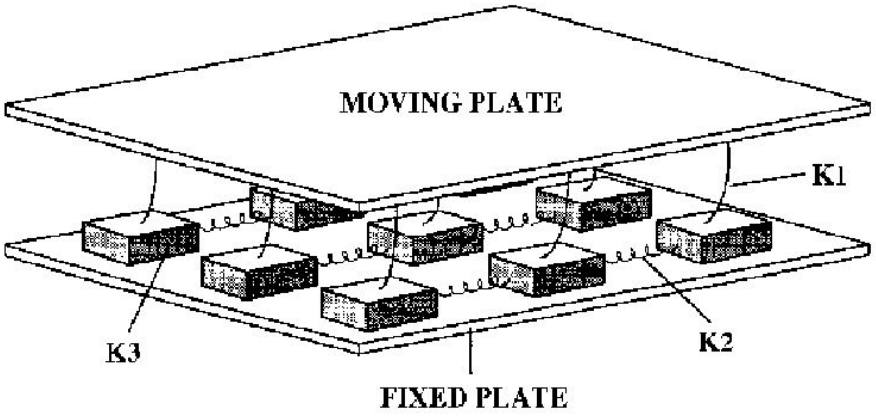
Hvad der styre et jordskælv, og hvornår de sker er ikke til at forudsige. Derfor har mange forskere på forskellige måde forsøgt at lave modeller til at forstå processen. En måde at betragte jordskælv på benytter sig af det der kaldes *Self-Organised Criticality* (SOC) (Bak et al. 1987). SOC siger at et system med mange frihedsgrader konstant befinder sig tæt på grænsen af stabilitet og selv meget små perturbationer (ændringer) af systemet kan føre til frigørelse af mere eller mindre energi. Det karakteristisk ved SOC modeller er at disse modeller producerer potens love (x^p) for hyppigheden af størrelsen af disse begivenheder.

I denne opgave benyttes en simpel model (OFC model, *Brown et al. 1991*) af et SOC system, som viser den karakteristisk effekt af disse modeller, i form af at den kan reproducerer potens fordelinger af begivenheders energiudladninger. Hvilken potens p de forskellige SOC modeller giver er ikke til at forudsige og dette er en klar kritikken af disse simple modeller. Yderligere er der mange der, lige som Game-of-life modellerne, benytter sig af regler som ikke altid kan føres direkte tilbage til en mere fysisk realistisk behandling af problemet.

I opgaven her går vi et lille stykke ud over den mest simple SOC model for jordskælv fra *Bak et al. 1998* og arbejder med en model der indeholder simpel fysik, og som samtidig opfører sig som et system der er i en kritisk tilstand. For at få lidt mere baggrund kan man med fordel skimme artiklen af *Brown et al. 1991*. De benytter en lidt anden realisering af den fysiske model end den der beskrives nedenfor.

Model:

Her beskrives en simpel model af jordskælv, der bygger på SOC. Det antages at der er to plader, som ikke indgår i selve skælvet, men som "forbindes" fysisk med hinanden via nogle bevægelige klodser. Disse klodser er forbundet med fjedre til hinanden og med en fjeder til den "øverste" plade ("Moving plate" i figuren)



Når den øverste plade forskydes i forhold til den nederste plade trækker den i de enkelte klodser via fjedrene. Når trækket er stort nok, sker det at den statiske friktionskraft mellem den nederste plade og nogle af klodserne overskrides af den samlede horisontale kraft. Disse ustabile klodser vil derfor bevæge sig i kraftens retning og nå en ny position hvor den samlede horisontale kraft igen er mindre end fiktionskraften. Denne forskydning af en klods ændre på fjeder-kræfterne på de klodser den forskudte klods er forbundet til og for nogle af disse kan den samlede kraft på dem nu overstiger friktionskraften. Disse nabo klodser bliver derved ustabile og forskydes igen. Således kan en dynamisk bevægelse i én klods pludselig skabe en kædereaktion og flere klodser forskydes indtil en ny stabil konfiguration er opnået. Dette antages for at repræsentere et jordskælv. Her kan man beregne både den frigivne energi og størrelsen af skælvet. Når hele systemet er stabilt stresses det igen og den processen gentages.

1D model:

Beregningerne kan laves simpelt som vist på figuren oven for, hvor de enkelte klodser kun er forbundet i en (x) retning via fjederen K_2.

For at lave denne model antages at modellen består af N klodser ($1xN$ array). Hver klods fordeles med en hvis afstand (l_0) mellem sig, og antages forbundet med en fjeder med en given fjederkonstant, $k_\pm$ (bemærk at for en given klods b_n vil k_+ have samme værdi som k_- for klodsen b_{n+1}.). Yderliger er der en fjeder K_1 som bruges til at trække i klodserne. Denne har fjederkonstanten k_p. Hver klods, b, antages at have en masse, M_b, en grundflade, A_b, og en friktionskoefficient μ_b per arealenhed (I første omgang kan du antage $k_\pm$, k_p, μ, A, og M er ens for alle klodser og fjedre, men det vil være en god ide at skriv koden således at hver klods kan have individuelle værdier - $k_{b,\pm}$, $k_{b,p}$, μ_b, A_b og M_b, hvor variationen kan ske via et mindre tilfældigt tal i forhold til grundværdien.).

Ud fra dette bliver friktionskraften som skal overskrides for at flytte klodsen b (så realt er det friktionskraftern $F_{b,frik}$ der er den fri variabel pr klods.):

$$F_{b,frik} = \mu_b M_b A_b$$

Den horisontale fjederkraft der påvirker klodsen b er givet ved:

$$F_{b,hor} = l_{p,b}\ k_p + l_+ k_+ - l_- k_-$$

Hvor $l_{p,b}$ er længden af den horisontale forskydning af top-planet i forhold til klodsen b. l_+ og l_- er afstandene til henholdsvis klodsen foran og bag klodsen b (afstand mellem masse centrum af blokkene), og k_+ og k_- er de respektive fjeder-konstanter (her kan simpelt antages at $k_+ = k_-$, men det vil være en fordel at tillade at de kan variere.).

Tip:

Afstandene mellem klodserne og forskydningen af top planet i forhold til klodsen b er givet ved

$$l_+ = l_0 + \delta_+ - \delta_b$$
$$l_- = l_0 + \delta_- - \delta_b$$
$$l_{p,b} = \delta_p - \delta_b$$

Hvor l_0 er start afstanden mellem klodserne og δ_b, δ_-, δ_+ er hvor meget henholdsvis klods b, b's venstre og højre nabo klodser er flyttet, og δ_p er forskydningen af top planet. Sætter vi $l_0 = 1$ kan vi omskrive formlen for den horisontale kraft oven over til

$$F_{b,hor} = (\delta_p - \delta_b)k_{b,p} + (1 + \delta_+ - \delta_b)k_+ - (1 + \delta_- - \delta_b)k_-$$

$$= \delta_p k_{b,p} - \delta_b(k_{b,p} + k_+ - k_-) + (1 + \delta_+)k_+ - (1 + \delta_-)k_-$$

Desuden ser vi at $F_{b,frik}$ er statisk for hver klods, således at vi kan definerer

$$F_{b,frik} = \mu_b A_b M_b = F_{0,frik}(1 + \sigma f_b)$$

Hvor $F_{0,frik} = 1$, f_b er individuel og normaltfordelt (altså Gaussisk fordelt med en middelværdi på 0 og spredning 1), og σ er den relative støj vi vil introducerer (f.eks. 5%).

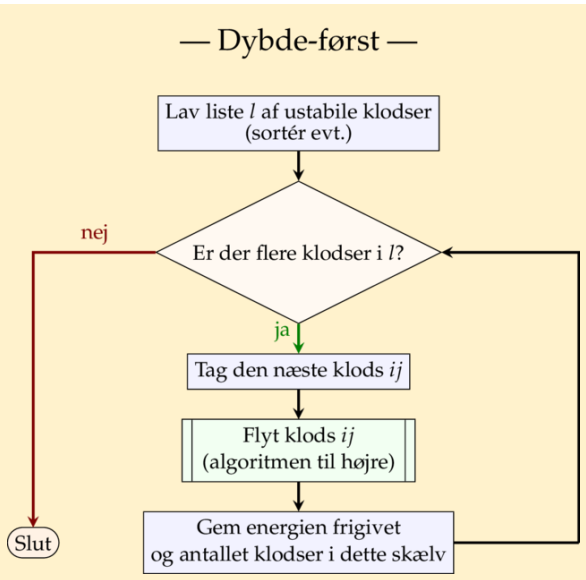
Med disse simplificerer vi beregningerne noget, og vi definere kræfter og fjederkonstanter i enheder af friktionskraften, og længder som relative længder.

Man driver systemet ved at ændre l_p med en lille smule, δl_p. Det kan gøres på to måder. Enten ændres l_p for alle klodserne på en gang, eller også så ændres l_p for en tilfældig valgt klods.

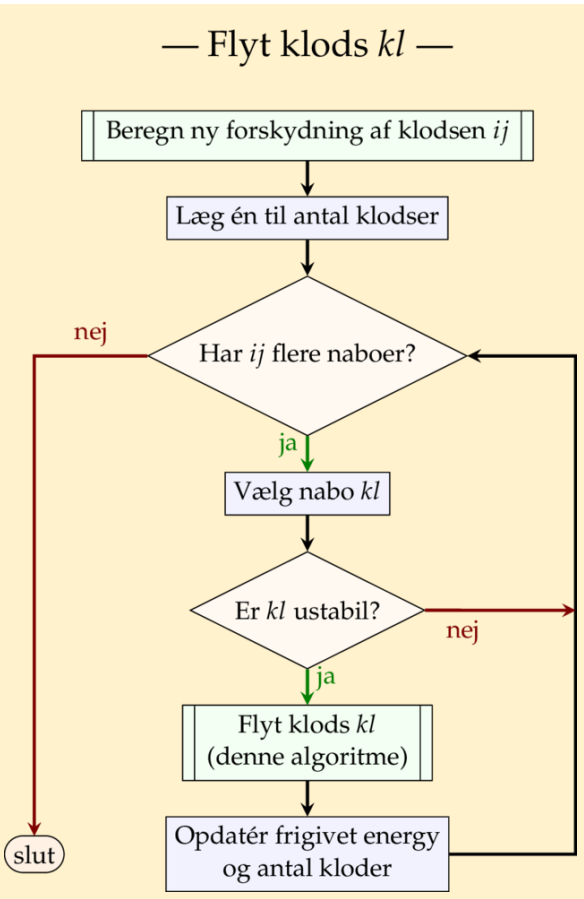
Tip:

To måder vi eksempelvis kan propagerer skælv på kan også betegnes *dybde-først* eller *bredte-først*. Som navnene antyder så går den første algoritme ud på at vi starter men én klods som udgangspunkt, mens den anden behandler alle klodser først. I begge tilfælde itereres algoritmen indtil alle klodser er stabile.

Dybde-først algoritmen



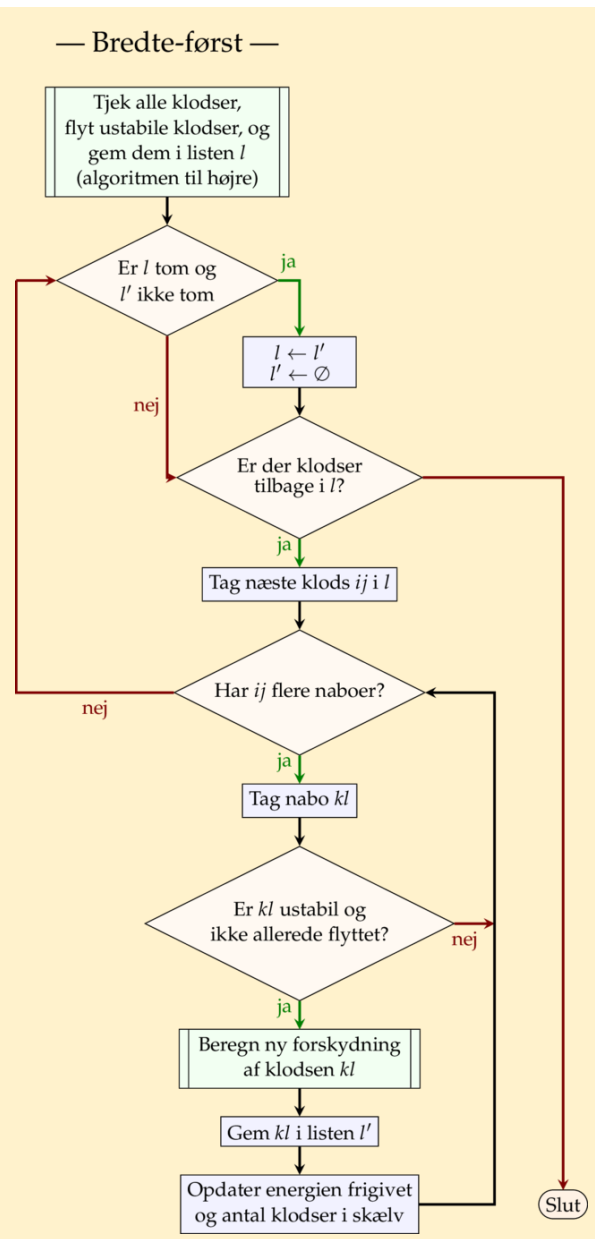
Toppen af dybde-først



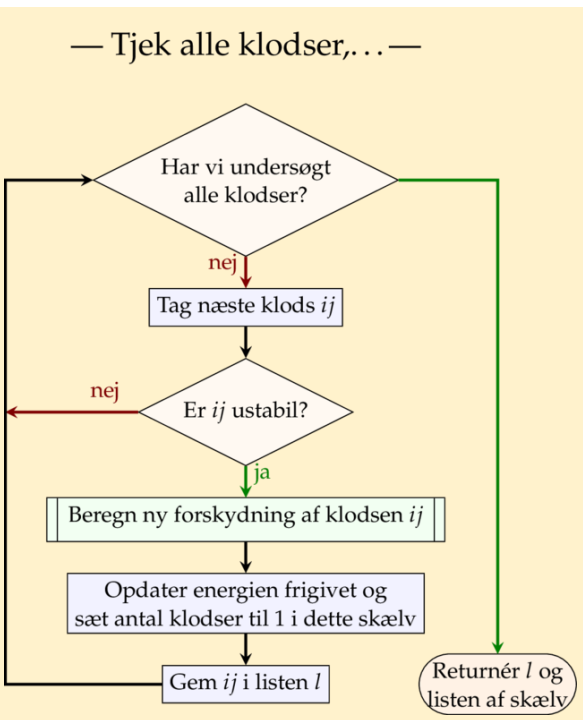
Rekursiv algoritme for at forskyde klodser

Figuren ovenover viser den overordnet struktur af dybde-først algoritmen. Bemærk at algoritmen til højre kalder sig selv rekursivt. I denne algoritme starter vi men én klods som er ustabil som forskydes. Dernæst kigger vi på dennes naboer og forskyder muligvis disse, og derefter kigger vi på deres naboer (den oprindelige klods' genboer), og således fortsætter vi rekursivt indtil alle naboer, genboer, genboers naboer, … er stabile. På den måde breder forskydninger sig ud som kegle toppe.

Bredte-først



Bredte-først algoritmen. Bemærk at vi hele tiden skaber en ny liste af "mødre" til kandidater til forskydning i næste skridt.



Starten af algoritmen hvori vi finder de første kandidater.

Figuren ovenover illustrerer hoved trækkene ved brede-først algoritmen. Læg mærke til at vi aldrig forskyder den samme klods mere end én gang. I denne algoritme starter vi med at kigge på alle klodser og bestemme dem som er ustabile. Derefter forskyder vi alle ustabile klodser én gang uden at rører ved naboer, samtidig med at vi noterer hvilke vi har flyttet. Så gentages denne procedure på naboer til dem der blev forskudt, indtil vi ikke længere har ustabile kandidater i vores liste. På den måde breder forskydninger sig ud som ringe.

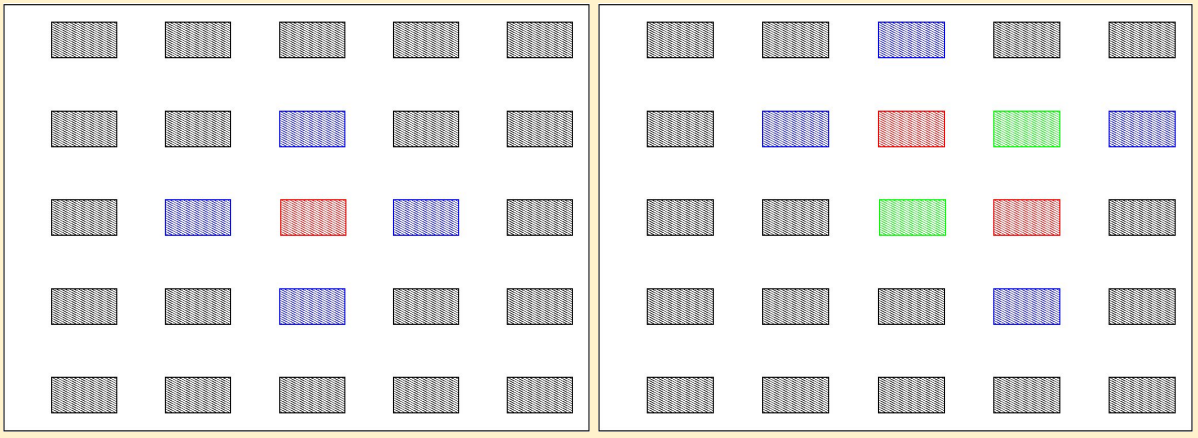
I begge algoritmer ovenover antages at *Beregn ny forskydning af klodsen ij* at bestemme den nye position for en given *enkelt* klods. Bemærk ligeledes at algoritmerne ovenover ikke stipulerer hvorledes information om skælv størrelse og energi udladning bogføres.

Disse to algoritmer er langt fra de eneste måder at implementerer skælv i modellen. Hvorvidt den ene eller anden eller en helt tredje algoritme benyttes er et selvstændigt valg.

Når det er gjort har de samlede kræfter på klodserne (klodsen) ændret sig og der testes om der er klodser for hvilke $F_{b,hor} > F_{b,frik}$. Hvis det er tilfældet flyttes denne klods til en ny position for hvilken den samlede kraft, $F_{b,hor}$, er faldet til $\alpha_b F_{b,frik}$. Her skal α_b have en værdi i intervallet $]0,1[$. Hvis den lille flytning δl_p sker for alle klodser samtidig er det vigtigt at bibeholde den oprindelige information om blokkene indtil alle er testet, for ikke at ændre noget i en forkert "tidsrækkefølge". Det samme gælder når man tester for ændringer skabt af de flytninger man lavede efter først gennemgang af systemet. Man kan med andre ord sige at vi arbejder med to tidsskalaer. En lang på hvilken systemet drives, og en kort som beskriver udviklingen af skælvet, med "oprydning" af ustabile klodser på samme tid inden for samme udviklingstrin.

Bemærkning:

En ting man skal passe på er hvordan man laver listen med naboklodser der skal undersøges efter en klods er flyttet. Figurerne nedenfor illustrerer problemet hvor samme klods senere i den hurtige tidsudvikling bliver markeret som kandidat til at flyttes, og derfor kan komme til at bidrage til processen flere gange indenfor et hurtigt tidsskridt.



Til venstre er den røde klods ustabil og skal flyttes. Det påvirker de 4 blå klodser som efterfølgende skal testet. I anden omgang, til højre, viser det sig at de 2 røde skal flyttes. Dette gør at de blå og grønne omkring dem efterfølgende skal testes. I den forbindelse flages de to grønne kloder ud af begge de røde klodser. Man skal derfor passe på med efterfølgende at beregne disse to gange, da det fejlagtigt kan bidrage til skælvets størrelsen og den frigivne energi.

Dette kan klares i 1D ved at sorterer listen af kritiske punkter og efterfølgende fjerne dupletter:

```
nb=sort(naboer);     % naboer er listen med nabo indexer, sort sorterer data
% fjerner flere forekomster af samme værdi fra nb vektoren
nbb=nb(1);    % ny vektor med sorterede værdier.
i1=1;
for i=2:length(nb)
    if nb(i1) ~= nb(i)
        nbb = [nbb,nb(i)];     i1=i;
    end
end
naboer=nbb;
```

Man kunne også bruge sorteringsalgoritmerne fra opgave 7, og her kun returnere en værdi i de tilfælde hvor der er flere forekomster af samme tal.

Det viser sig dog hurtigt at dette bliver meget tidskrævende når der er mange klodser der skal testes på det næste tidsskridt. Det kan derfor betale sig at indføre et array af samme størrelse som det data grid der arbejdes med. Initialiserer dette til 0 og ændre værdien i (i,j) positionen til 1 første gang der skal arbejdes på den. Hvis værdien i (i,j) allerede er sat til 1, ved man at den klods er opdatet.

Ændringen af den givne klods position, δ_b, fås ved at beregne den ændring i positionen som kræves for at den samlede kraft på klodsen når den ønskede værdi:

$$\alpha_b F_{b,frik} = k_p(l_{p,b} - \delta_b) + k_+(l_+ - \delta_b) - k_-(l_- + \delta_b)$$

Efter lidt omskrivning finder man at

$$\delta_b = -\frac{\alpha_b F_{b,frik} - k_p l_{p,b} - k_+ l_+ + k_- l_-}{k_p + k_+ + k_-}$$

Som det ses, så ændre forskydningen med δ_b længderne l_+ og l_- med δ_b og dermed bliver kræfterne på nabo klodserne også ændret. Vi skal derfor tjekke om naboerne er blevet ustabile - den samlede kraft på dem overstiger deres friktionskraft - og derfor skal flyttes.

For at løse systemet må vi antage bestemte randbetingelser. Her er 2 muligheder. De kan antages hele tiden at ligge fast og dermed være med til at forankre hele systemet så de holdes inden for den samlede længde. Eller man kan antage at systemet er periodisk, så alle klodser i systemet opfører sig ens.

Tip:

Periodiske randbetingelser kan give et problem alt efter hvordan man implementerer ovenstående. Typisk skal man for klods *1* og *N* have adgang til information om deres naboer, hvoraf den ene ligger i den modsatte ende af ens data-vektor.

For at få fat i dette index kan man simpelt gøre:

```
if i==1
```

```
        i_op=2; i_ned=N;
    elseif i==N
        i_op=1; i_ned=N-1;
    else
        i_op=i+1; i_ned=i-1;
    end
```

Hvor *i* er indexet for den aktive klods og *i_op* og *i_ned* er indeksene for naboer i et 1D periodisk system.

Energi beregning:

For at beregne den energi der er frigivet ved flytningen af en klods, integreres den kraft ændring som finder sted over den vejlængde klods flytter sig.

$$E_b = \int_0^{\delta_b} F_{b,hor}\, dx$$

Da modellen arbejder med fjedre, er integralet en beregning af kraft ændringen i de 3 fjedre mellem de to positioner, før og efter flytningen og den frigivne energi er det tabte arbejde fjedrene har udført for at bringe det til en lavere energitilstand,

$$E_b = k_p((l_{p,b} - \delta_b)^2 - l^2_{p,b}) + k_+((l_+ - \delta_b)^2 - l^2_+) - k_-((l_- + \delta_b)^2 - l^2_-)$$

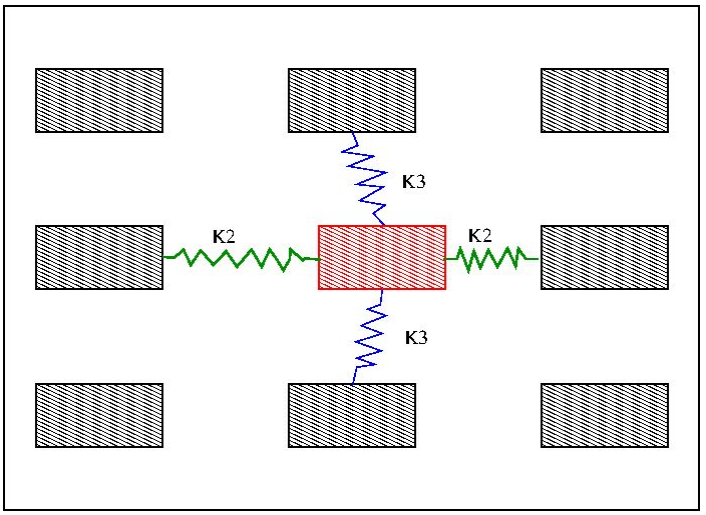
En anden proxi for dette kan simpelt fås ved at antage at man hele tiden har skulle arbejde mod friktionskraften:

$$E_b = F_{frik,b}\ \delta_d$$

2D model:

Modellen ovenfor er for simple til at lave mere komplicerede jordskælv, men den indeholder den generelle algorithm for at virke. For at gøre det mere realistisk udvides modellen til 2 dimensioner. Dette gøres ved at tilføje endnu et sæt fjedre, med fjedre konstant k_3, mellem klodserne, så de også bliver forbundet på tværs af den tidliger bevægelsesretning. Denne udvidelse gør at man ikke længere kan regne i simple længder, men bliver nød til at indfører vektorer langs klodsernes forbindelslinjerne, for derved at kunne beregne de retnings afhængige krafter på klodsen. Dette gør at klodsen ikke bare bevæger sig i "*x*"-retningen når den flyttes til en ny stabil position, men den er fri til at flytte sig i både "*x*" og "*y*" -retningerne. Dette gør det svære at beregne dens nye position.

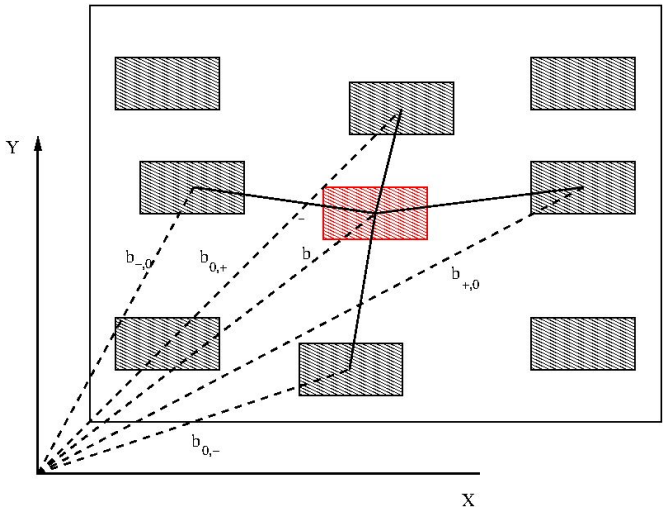
Friktionskraften er den samme som ovenfor, men dens retning er modsat af den samlede kraftvektor, som nu skal beregnes ud fra de respektive afstands vektorer (mellem masse centrum af blokkene) og de respektive fjederkonstanter, som indikeret i billedet nedenfor for en enkelt klods. Det antages at kraften fra den overliggende plade hele tiden er i "*x*" -retningen.



Hvis vi med $\vec{b}$ betegner stedvektoren for den klods vi undersøger og med $\vec{b}_{-,0}$, $\vec{b}_{+,0}$, $\vec{b}_{0,-}$ og $\vec{b}_{0,+}$ forstår henholdsvis stedvektorene for naboen til venstre, højre, nedenunder og ovenover, så kan vi skrive

$$\vec{F}_{b,hor} = k_p l_{p,b} + k_{-,0}(\vec{b}_{-,0} - \vec{b}) + k_{+,0}(\vec{b}_{+,0} - \vec{b}) + k_{0,-}(\vec{b}_{0,-} - \vec{b}) + k_{0,+}(\vec{b}_{0,+} - \vec{b})$$

Hvor $l_{p,b}$ er afstand til top planet i "x"-retningen.



I figuren repræsentere de stiblede linier de respektive stedvektorer for de 5 blokke som indgår i beregningen af kraften på den røde blok. De tykke sorte linier er de afstande som benyttes i regneudtykket for $\vec{F}_{b,hor}$ der er givet over figuren.

Tip:

Sætter vi som ovenover $l_0 = 1$ er stedvektoren for en klods med start position (i,j) givet ved

$$\vec{b}_{ij} = (i + \delta_i, j + \delta_j)$$

Afstandsvektoren mellem to klodser (i,j) og (k,l) er derfor

$$\vec{b}_{kl} - \vec{b}_{ij} = (k + \delta_k, l + \delta_l) - (i + \delta_i, j + \delta_j) = ([k - i] + [\delta_k - \delta_i], [l - j] + [\delta_l - \delta_j])$$

For nabo klodser ses at $(k - i), (l - j) \in [-1, 0, +1]$ og vi kan således nøjes med at gemme ændringerne i positionerne $\delta_{ij} = (\delta_i, \delta_j)$.

Beregning af ny position

Friktionskraften på klodsen b er modsat rettet af den horisontale kraft $\vec{F}_{b,hor}$ så at

$$\vec{F}_{b,frik} = -F_{b,frik}\,\vec{F}_{b,hor}/|\vec{F}_{b,hor}| = -\mu_b A_b M_b \vec{F}_{b,hor}/|\vec{F}_{b,hor}|$$

Den tilbageværende kraft, efter valg af α er da

$$\vec{F}_{b,res} = \alpha F_{b,frik}\vec{F}_{b,hor}/|\vec{F}_{b,hor}|$$

Vi skal nu finde $\vec{\delta}_b$ så at

$$\vec{F}_{b,hor} + \alpha\vec{F}_{b,frik}\vec{F}_{b,hor}/|\vec{F}_{b,hor}| = 0$$

$$\vec{B}(\vec{\delta}_b) = \vec{F}_{b,hor}(1 + \alpha F_{b,frik}/|\vec{F}_{b,hor}|) = 0$$

Hvis vi antager at den resulterende krafts retning ikke ændres under forskydningen, hvilket svarer til at $|\vec{\delta}_b| \ll l$, så er $\vec{\delta}_b$ parallel med $\vec{F}_{b,hor}$ så vi kan skrive

$$\vec{\delta}_b = \delta_b \vec{F}_{b,hor}$$

og problemet består i at finde δ_b så at nulpunktet findes. Her gøres en lille fejl i det kraften kan ændre retning undervejs i forskydningen, men så længe $|\vec{\delta}_b| \ll l$ er denne fejl lille. På grund af de mange komponenter der indgår i $\vec{F}_{b,hor}$ som alle er afhængige af δ_b er det ikke trivielt at finde δ_b, så vi kan ty til numeriske metoder til at løse ligningen for $\vec{B}$, hvor vi typisk definerer konvergens som at $|\vec{B}(\vec{\delta}_b)| < \varepsilon$. Her kan man bruge forskellige metoder - f.eks. Newton-Raphson nul-punkts metode, gradient nedstigning, eller lignende.

Da $\vec{B}(\vec{\delta}_b)$ ikke er en lineær funktion og vi ikke vil implementere en kompliceret metode her, vælger vi en lettere tilgang via en simpel iterations model. Ud fra ovenstående ved vi at hvis klodsen bevæges kort i den resulterende krafts retning vil den ende et sted med en laver værdi for $|\vec{B}(\vec{\delta}_b)|$. Ud fra test kan man vælge $\delta_b = 0.1|\vec{B}(\vec{\delta}_b)|$ for hvert step, hvilket cirka halverer fejlen i hvert iterations skridt.

Energi frigivelse

Som oven for er den frigivne energi givet ved at integrere ændringen i fejder arbejdet mellem start og slut position

$$E_b = \int_0^{\delta v_b} F_{b,hor}\, dv$$

Dette kan løses på samme måde som beskrevet under 1D modellen. Regneudtrykkene er bare mere komplicerede for den del der benytter de mange fjederkræfter, hvor det her er længde ændringen af de 5 fjedre som skal benyttes.

Programerings opgave:

Selve programmerings opgaven går ud på at implementer de to modeller der er beskrevet ovenfor. Her skal man let kunne ændre de beskrevne parametre og vælge mellem om der benyttes en 1D eller 2D model. Det er valgfrit om man driver alle klods på én gang, eller en tilfældigt klods ad gangen. Der er også frit at vælge mellem periodiske og låste randbetingelser.

> Tip:
>
> Forsøg så vidt muligt at teste enkelte beregninger. F.eks. kan man definerer én klods med en given friktionskraft og én fjeder som man trækker i. På den måde kan man tjekke sin løsning af ligningen $\vec{F}_{b,hor} + \alpha \vec{F}_{b,frik} \vec{F}_{b,hor} / |\vec{F}_{b,hor}| = 0$. Derefter kan man komplicerer problemet ved at tilføje 2 eller 4 fjedre til klodsen.
>
> Husk også at dokumenterer koden ved hjælp af kommentarer så det er klart for alle hvad der sker hvor.

Som resultatet af en kørsel, skal det være muligt at vise hvordan et fiktivt jordskælv breder sig i rummet under en enkelt energiudløsning, hvor meget energi der er frigives som funktion af tid, og hvordan fordelingen af antallet af skælv med en given størrelse ser ud med de benyttede parameter.

Det kan være en fordel at lave de grafiske data efter selve beregningen er færdig. For eksempel kan man skrive parametre (tidspunkt, størrelse, total energi udladning, osv.) for hvert skælv til en fil og så derefter indlæse denne fil med henblik på en grafisk repræsentation.

Da dette kan være regne tungt at lave tilstrækkelige lange kørsler med et passende antal klodser (omkring 100x100), er det klart en fordel at prøv at optime beregningerne.

> Tip:
>
> Start med mindre systemer (f.eks. 10x10 eller endda 1x10) for at tjekke at din løsning opfører sig som du forventer.
>
> På den måde kan du også bestemme hvor mange tidskridt der skal til før en klodserne begynder at forskydes. Den viden kan så bruges til at sætte en startværdi af l_p (for eksempel et sted mellem 0.5 og 0.9), sådan at din kode ikke skal tage for mange tidsskridt inden der sker noget. Den øvre grænse af l_p kan bestemmes ved $l_p k_P < F_{b,frik}$ for alle klodser.

Parametre:

Her lister vi de variable parameter der er i problemet og fornuftige begyndelsesværdier:

Alt regnes her uden fysiske dimensioner og resultatet kan derfor ikke direkte overføres til fysiske systemer.

Variable	**Beskrivelse**	**Startværdi**	**Variation**
$F_{0,frik}$	Friktionskraft (μMA)	1	5%
l_p	længde til trækplade	0	0
l_0	afstand mellem klodser	1	5%
k_P	Fjederkonstant / trækplade	1	5%
$k_{\pm}, k_{\{\pm,0\},\{\pm,0\}}$	Fjederkonstant / klodser	1	5%
α	Brøkdeldel af normalkraft efter klods flytning	0.75	5%
$N_{x,y}$	Antal blokke i model	100,100	efter ønske
ε	konvergenes kriterie for 2D model ifb med klods flytning	0.0001	

Rapport:

Ud over programmeringsdelen af opgaven, skal der skrives en *kort* rapport. Rapporten skal virker som en brugervejledning der med passende information om de punkter som er beskrevet nedenfor. Dette er med til at vise at du har en overordnet forståelse af selve problemet og kan forklare hvordan dit programmet kan bruges til at løse problemet.

Rapporten skal skrives så en studerende med et lignende uddannelsesniveau, men uden programmerings erfaring i *Matlab*, ud fra rapporten er i stand til at benytte programmet, forstå resultatet og have en vis ide om hvor troværdigt resultatet er. Her er de punkter der bør indkluderes i rapporten:

- Titel side.
    - Titel på projektet.
    - Navn.

    - KU userid.
- Introduktion til problemet.
    - Hvilket problem søges løst, og hvad er løsningsmodellen.
    - Hvad er den basale ide i modellen.
    - Hvad er formålet med rapporten.
- Beskriv programmet.
    - Hvordan er det opbygget
        - Flow-diagram etc.
        - Hvilke unit test er med.
        - **N.B.:** Direkte citater fra koden er *ikke* velset — kun for så vidt at det handler om helt særlige "smarte" finurligheder og at koden forklares i teksten.
    - Hvordan virker det.
    - Hvordan bruges det.
        - Hvordan vælges begyndelsesbetingelser (valg af frie variable).
        - Hvordan køres de forskellige modeller.
        - Hvordan gemmes data og hvordan plottes de efterfølgende.
        - Hvilke begrænsninger er der mht koden.
    - Dokumentation af at programmet virker.
        - Inkluder forskellige figurer der viser situationer fra tidsudviklingen af systemet.
- En opsummering af opgaven.
    - Dit perspektiv på opgaven og dens resultat.

Rapporten kan skrives på dansk eller engelsk. Den skal afleveres som en PDF fil.

Inden du afleverer opgaven på Digital eksamen siden, skal alle vigtige filer samles i en zip fil, *projekt.zip*, der skal indeholde rapporten og alle de *Matlab* filer mm der benyttes til eksekvering og testning af projektet.

Både rapport og kode skal laves *individuelt*.

Referencer:

- Bak, P., and Tang, C.; Earthquakes as a self-organized critical phenomenon, *J. Geophys. Res.*, 94(B11), 15635–15637, 1989.
- Bak, P., Tang, C., and Wiesenfeld, K.; Self-organized criticality, *Phys. Rev.* **A38**, 364-374, 1988.
- Brown, S., Scholz, C. H., Rundle, J. B.; A simplified spring-block model of earthquakes , Geophysical Research Letters 18(2):215-218, 1991

- Spring-Block Models of Earthquake Dynamics Master thesis
- OFC Slider Block Model Java kode af Olami-Feder-Christensen model af jordskælv
- Earthquake Machine: Demonstration of the 2-block Model Video

- The Olami-Feder-Christensen Model on a Small World topology - præsentation
- A cellular-automata, slider-block mode for earthquakes I. Demonstration of chaotic behaviour for a low-order system Artikel