Programmeringsuppgift 2
Game of Life

Innan du börjar:
För att kunna lösa uppgiften nedan behöver du ha kunskap om

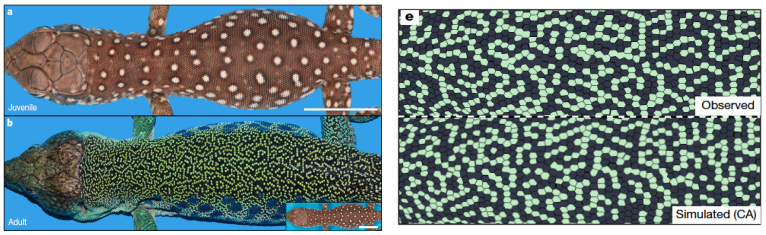
1. Hur man hanterar matriser i Matlab
2. Grundläggande programmeringsstrukturer som `for`- och `while`-slingor samt `if`-satser.
3. Hur man skriver egna funktioner.

Gör de obligatoriska förberedande uppgifterna i Matlab Grader för Programmeringsuppgift 2 innan du påbörjar arbetet med uppgifterna nedan. Uppgifterna görs individuellt och ska vara klara senast den 8/10-18.

Redovisning:
Funktioner och huvudkod skicka in via Canvas senast den 7/10-18 kl 15.00 och redovisas sedan muntligt vid laborationstillfället den 8/10-18. Tider för muntlig redovisning bokas i Canvas.

Inledning

En cellulär automat är en modell i form av ett rutnät av celler, där cellerna har olika tillstånd. Tillståndet för en cell förändras enligt ett antal levnadsregler som beror på tillstånden hos de närliggande cellerna. Biologiska mönster kan förstås genom cellulär automation, t ex hur fläckarna på en leopard eller ränderna på en zebrafisk växer fram över tid. En studie i tidskriften Nature [1] visar med hjälp av numeriska beräkningar som jämförs med observationer hur pärlödlans ryggmönster - ett labyrintliknande grönt och svart mönster - produceras genom cellulär automation med hexagonallik struktur. I figur 1 ser vi illustrationer från journalartikeln [1] där studien återfinns.



Figur 1: Till vänster: Pärlödlans ryggmönster. Till höger: Observerat och simulerat tillstånd.

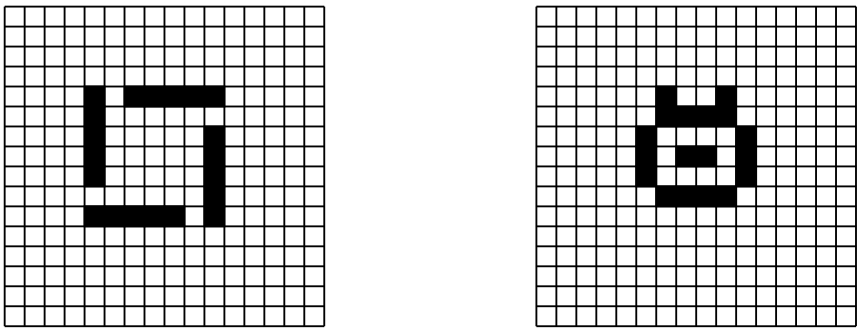
I denna uppgift ska du implementera ett spel som är en cellulär automation med schackrutigt nät. Spelet kallas Game of Life och uppfanns av John Conway 1970. Givet en startuppställning av en koloni av celler i en tvådimensionell värld och regler för hur celler föds och dör ska du simulera hur kolonin utvecklas.

Levnadsregler för cellerna:

- En cell som har färre än två grannar dör av ensamhet.
- En cell som har två eller tre grannar överlever till nästa generation.
- En cell som har fler än tre grannar dör av överbefolkning och svält.
- På en tom plats där det finns exakt tre grannar föds en ny cell till nästa generation.
- Cellerna föds och lever bara i den simulerade världen, inte utanför den.

I den tvådimensionella världen har varje cell plats för åtta grannar. Grannar kan finnas på direkt intilliggande platser horisontellt, lodrätt eller diagonalt. Hur en cell ska förändras till nästa steg skall beräknas innan någon annan cell förändras.

Den tvådimensionella världen med celler kan representeras med en matris med nollor och ettor. Elementen i matrisen representerar platser i kolonin. Ett element med värdet noll (0) representerar ingen (eller en död) cell och ett element med värdet ett (1) representerar en levande cell. Se Figur 2.



Figur 2: Exempel på starttillstånd. Svarta rutor betecknar levande celler och vita rutor betecknar döda (=inga) celler. Båda tillstånden kan representeras med en 16×16-matris med ettor för de svarta rutorna och nollor för de vita rutorna.

Simulering av kolonin med celler

1. Skriv en funktion `Antalgrannar.m` som givet en plats i den två-dimensionella världen räknar ut antalet grannar som innehåller levande celler. Indata till funktionen ska vara matrisen som definierar cell-kolonin och index för den aktuella platsen. Utdata från funktionen ska vara antalet grannar.
2. Skriv en funktion `Levnadsregler.m` som för en given plats i kolonin implementerar levnadsreglerna. Funktionen ska returnera den uppdaterade statusen för platsen som innebär antingen en levande/ny cell (1) eller död/ingen cell (0).

   Innan du skriver funktionen, fundera på vad som är lämpliga indata till funktionen.

3. Skriv ett huvudprogam som med hjälp av de två funktionerna ovan simulerar kolonins utbredning i tiden och ritar ut resultaten på skärmen. Kom ihåg att programmet måste räkna ut alla förändringar innan det går över till nästa generation. Simulera olika uppsättningar av celler i världen. Se figur. 2 för två exempel på starttillstånd. Kommer kolonin att expandera, dö ut eller nå ett stabilt tillstånd?

Tips 1: Gör en utökning av den tvådimensionella världen med tomma platser så att varje plats i den inre domänen (den simulerade världen) har 8 grannar.
Tips 2: Man kan använda MATLAB-kommandot `spy()` för att rita ut resultaten på skärmen, se vidare `help spy`.

En större koloni att simulera: Puffer train

4. Ladda ner filen `puffer.mat` från kurshemsidan. Filen innehåller en matris `A` av storlek 170×400 som är ett starttillstånd som vid simulering ger upphov till ett specifikt mönster som kallas för Puffer train [1]. Modifiera ditt huvudprogram så att du läser in matrisen i i början av programmet med kommandot `load puffer.mat`. Simulera sedan utvecklingen av detta starttillstånd. Identifiera några strukturer som är stationära, oscillerar, eller förflyttar sig utan att ändra form. Vad händer med utvecklingen av kolonin om du ändrar på starttillståndet något litet, verkar starttillståndet vara känsligt för små förändringar? Skapa gärna egna starttillstånd och testa vad som händer!

Tips 3: Om du inte gjort detta redan: modifiera ditt huvudprogram så att det kan hantera en matris av godtycklig storlek. Kommandot `[N,M] = size(A)` lagrar antalet rader i matrisen `A` i variabeln `N` och antalet kolumner i variabeln `M`.

Referenser

[1] L. Manukyan et. al., *A living mesoscopic cellular automaton made of skin scales*, Nature **544**, pp.173–179 (2017).

[1] `https://en.wikipedia.org/wiki/Puffer_train`