Simulering av differenslikninger

Programmering i Java med eksempler

Forelesning uke 39, 2006

MAT-INF1100

Løsning av differenslikninger i formel

Mulig for lineære likninger med konst. koeff. og enkelte inhomogeniteter.
Eksempel: (b, c er konstante)

$$x_{n+2} + bx_{n+1} + cx_n = \cos(n), \quad x_0 = b_0, \quad x_1 = b_1.$$

1. Løs karakteristisk likning $r^2 + br + c = 0$
2. Bestem hvilket tilfelle vi har (2 reelle røtter, sammenfalne røtter, komplekse røtter) og sett opp generell $x_n^{(h)}$.
3. Finn en partikulærløsning $x_n^{(p)}$, dersom mulig
4. Bestem de valgbare konstantene i $x_n^{(h)}$ slik at $x_n^{(h)} + x_n^{(p)}$ oppfyller initialbetingelsene $x_0 = b_0$, $x_1 = b_1$

Hva kan løses i formel; eksempler

Likning	løsning
$x_{n+2} + 2x_n = n^2$	ja
$x_{n+2} + 2\sin(n)x_n = 0$	nei
$x_{n+2} + 2x_n = \sqrt{1+n}$	nei
$x_{n+2} + (x_n)^2 = 0$	nei
$x_{n+3} + 3x_{n+1} + x_n = e^n$	ja (ikke i pensum)
$x_{n+1} + (n+1)x_n = 0$	ja, spesialtilfelle

Tallsvar fra differenslikninger

Vi har tre naturlige måter å **beregne** løsninger av differenslikninger

1. Kode formelløsningen.
   Grei jobb på kalkulator eller datamaskin
2. Kode stegene i prosedyren beskrevet på forrige lysark.
   En litt krevende oppgave på vårt nivå– langt program
3. Direkte simulering.
   Enkelt og generelt

De to første kan bare brukes når en formelløsning kan finnes.
Vi skal se mer på det siste alternativet.

Differenslikning av orden 2

$$x_{n+2} + bx_{n+1} + cx_n = 0, \quad x_0 = b_0, \quad x_1 = b_1, \quad (1)$$

Klassifiering: Lineær, av annen orden, konstante koeffisienter, homogen $\Rightarrow$
Vi kjenner formler for generelle løsninger

I stedet: direkte simulering av (1)
Program skal

- Spørre etter og lese b, c, b_0, b_1 og maks. n (**nmax**) fra skjerm
- Løse differenslikning
- Skrive resultat til fil
  Skal noe gjøres med svaret, feks. plotting, kan vi ikke bare la tallene "rulle" over skjermen

Lesing av parametere

Vi bruker **easyIO**, beskrevet i *Rett på Java*, kap. 3

Spørsmål

Deklarasjon av objekt og utskrift

```
Out skjerm=new Out();
....
skjerm.outln("gi koeff. b og c");
```

Innlesning

Deklarasjon av objekt og innlesning

```
In les=new In();
...
b=les.inDouble();c=les.inDouble();les.inLine();
```

Skriving til fil – lett!

Omhandlet i *Rett på Java*, kap. 9
Dersom vi vil åpne og skrive på en fil **solv.dat**, brukes filnavn som argument i deklarasjon av ny **Out**

```
Out utf=new Out("solv.dat");
...
utf.out(x);....
....
utf.close();
```

Filen må lukkes etter at den er ferdig skrevet.
Vi bruker enkel uformattert utskrift – ikke pent, men det virker.

Digresjon; omdirigering av output

I Linux (og alle andre UNIX varianter) finnes det et enkelt alternativ til å åpne en fil i programmet og skrive på denne.

Dersom vi bruker **Out skjerm=new Out();** vil kommandoen

linux>java Ordto

der **Ordto** er klassenavnet, føre til at resultatene skrives i terminalvinduet. Vi kan redirigere resultatstrømmen ved

linux>java Ordto > res.dat

Nå opprettes en fil, **res.dat**, og resultatene skrives der.

Linux tilbyr mange liknende varianter for fleksibel behandlig av input/output. Vi legger ikke vekt på dette i MAT-INF1100.

Etterbehandling av fil

Vi vil se på den på skjermen under Linux. Skriv

less solv.dat

Vi vil bytte navn til **res.dat**

mv solv.dat res.dat

Eller fjerne fila

rm solv.dat

Et vanlig ønske er å framstille datane grafisk. I dag viser jeg bare resultatet, men sier ikke hvordan!

Selve beregningen

Den numeriske kjernen i løsningen er en løkke.

Naturlig ide: bruk av array

```
...
double[] x;
...
x=new double[nmax+1];
/* gir indekser 0....nmax */
...
for(i=2;i<=nmax;i++) {
x[i]=–b*x[i–1]–c*x[i–2];
...};
```

Array er unødvendig bruk av plass. Viktig å unngå det ?
Neppe her, men i andre, liknende, sammenhenger kan det være viktig.

Uten bruk av arrayer

La oss bruke **x** og **xprev** for x_n og x_{n-1}.
Verdier skal være riktige ved hver slutten av hver løkke

Bom 1

```
for(i=2;.....) {
x=–b*x–c*xprev;
xprev=x;};
```

Galt! Hvorfor ?

Bom 2

```
for(i=2;.....) {
xprev=x;
x=–b*x–c*xprev;};
```

Galt igjen! Hvorfor ?

Bruk av mellomlagring

Dette virker

Vi bruker en ekstra variabel **tmp**

```
for(i=2;.....) {
tmp=x;
x=–b*x–c*xprev;
xprev=tmp;};
```

Nå kan vi sette sammen programmet.

Filer **andreor.java** og **andarray.java**

Observasjoner

-Tenk nå igjennom hvordan formelene i Kalkulus kan kodes slik antydet tidligere. Det blir et mye lengre program enn **andreor.java**.

-Det er mulig å trikse seg unna mellomlagring (oppgave i Kompendium)

Test-eksempel; Fibonacci følgen

$b = c = -1$, $b_0 = b_1 = 1 \Rightarrow$

$$x_{n+2} - x_{n+1} - x_n = 0, \quad x_0 = 1, \quad x_1 = 1$$

Svar: 1, 1, 2, 3, 5, 8, ...

Kommando og dialog med programmet

linux>java Ordto
gi koeff. b og c
–1 –1
gi initialbet. b0 og b1
1 1
gi antall ledd (max n)
8

Fibonacci-test; resultat

```
linux>cat solv.dat
0 1.0
1 1.0
2 2.0
3 3.0
4 5.0
5 8.0
6 13.0
7 21.0
8 34.0
```

God regel: program testes på eksempel med kjent svar

Nytt eksempel

Likner på det i kompendiumet, 4.2.
$b = -31/3$, $c = 10/3$, $b_0 = 1$, $b_1 = 1/3 \Rightarrow$

$$x_{n+2} - \frac{31}{3}x_{n+1} + \frac{10}{3}x_n = 0, \quad x_0 = 1, \quad x_1 = \frac{1}{3}$$

Karakteristisk polynom=0

$$r^2 - \frac{31}{3}r + \frac{10}{3} = 0 \Rightarrow r_1 = \frac{1}{3}, r_2 = 10$$

Løsning som oppfyller initialbetingelser

$$x_n = \left(\frac{1}{3}\right)^n$$

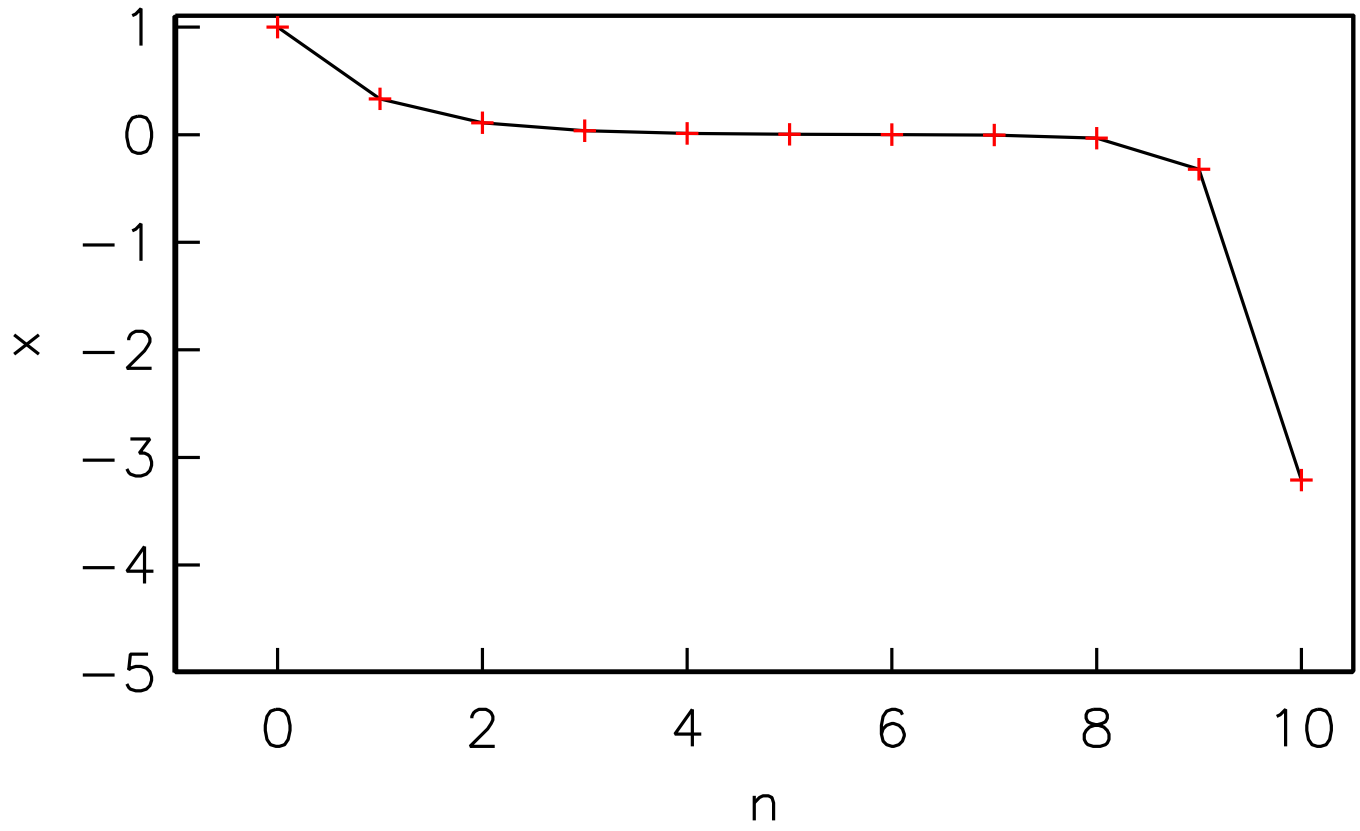
Simulering med $b = -31/3$, $c = 10/3$

Dialog med programmet

gi koeff. b og c
–10.33333333 3.33333333
gi initialbet. b0 og b1
1.0 0.33333333
gi antall ledd (max n)
12

Løsning bør være nær $x_n = (\frac{1}{3})^n$.
Spesielt $x_n \to 0$ når $n \to \infty$

Graf



Det ser bra ut fram til $n = 8$. Da går det helt galt!
Hva er feil ?

Hva går galt

Feilkilder i simulering

1. Initialbetingelser gis inn unøyaktig, feil er omlag 10^{-9}
2. Koeffissienter gis også inn feil
3. Avrundingsfeil påvirker utregning av hver ny x_n

pkt. 1 gir litt gale initialbetingelser; løsning modifiseres ala

$$x_n = (1+\delta)\,(1/3)^n + \epsilon\,(10)^n\,,$$

der ϵ og δ er av orden 10^{-9}. Nå vil $x_n \to \pm\infty$ når $n \to \infty$. Ved omtrent $n = 9$ blir det andre ledd av orden 1.

Uansett hva vi gjør vil dette fenomenet dukke opp for tilstrekkelig stor n pga. avrunding i pkt. 1–3

Enkel populasjonsdynamikk

Vanlig enkel lov for populasjonsvekst

$$y_{n+1} = \mu y_n$$

der n teller generasjoner el.
Klassifisering:Lineær, første orden, konstant koeff., homogen.
Løsning (der y_0 er gitt)

$$y_n = \mu^n y_0$$

$\mu > 1 \Rightarrow y_n$ vokser over alle grenser
$\mu < 1 \Rightarrow$ populasjonen dør ut

For enkel modell som igorerer bla.a. begrensninger i tilgang på ressurser.

Ikkelineær modell

Negativ konsekvens av konkurranse antas proporsjonal med kvadratet poulasjonen (sjansen for at to individer treffes tilfeldig og slåss om mat etc. er proposjonal med kvadratet)

$$y_{n+1} = \mu y_n - B y_n^2$$

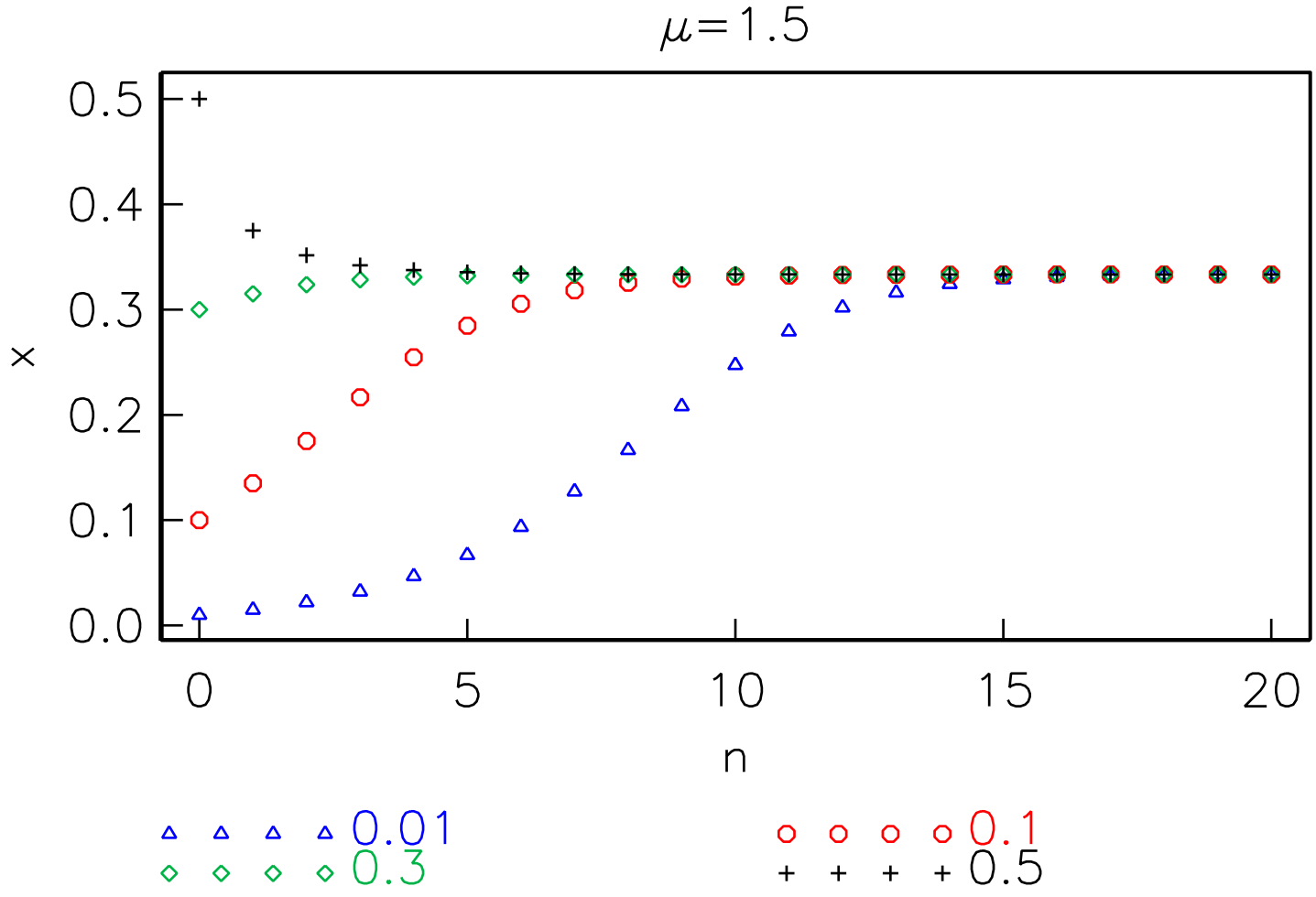
Bytte av variabel $y_n = \mu x_n / B$:

$$x_{n+1} = \mu x_n (1 - x_n)$$

Klassifisering: Ikkelineær, første orden, homogen(?!).
Kan ikke løses i formel (hvorfor?), men enkel å simulere
Interessante tilfeller $\mu > 1$.

Program **ulin.java**

$\mu = 1.5, \quad x_e = 1/3$



Kurver merket med x_0. Konvergens (ser det ut som).

Likevektsløsning

Det synes som x_n har en grense, x_e, når $n \to \infty$.
Det finnes løsning $x_n = x_e$=konstant:

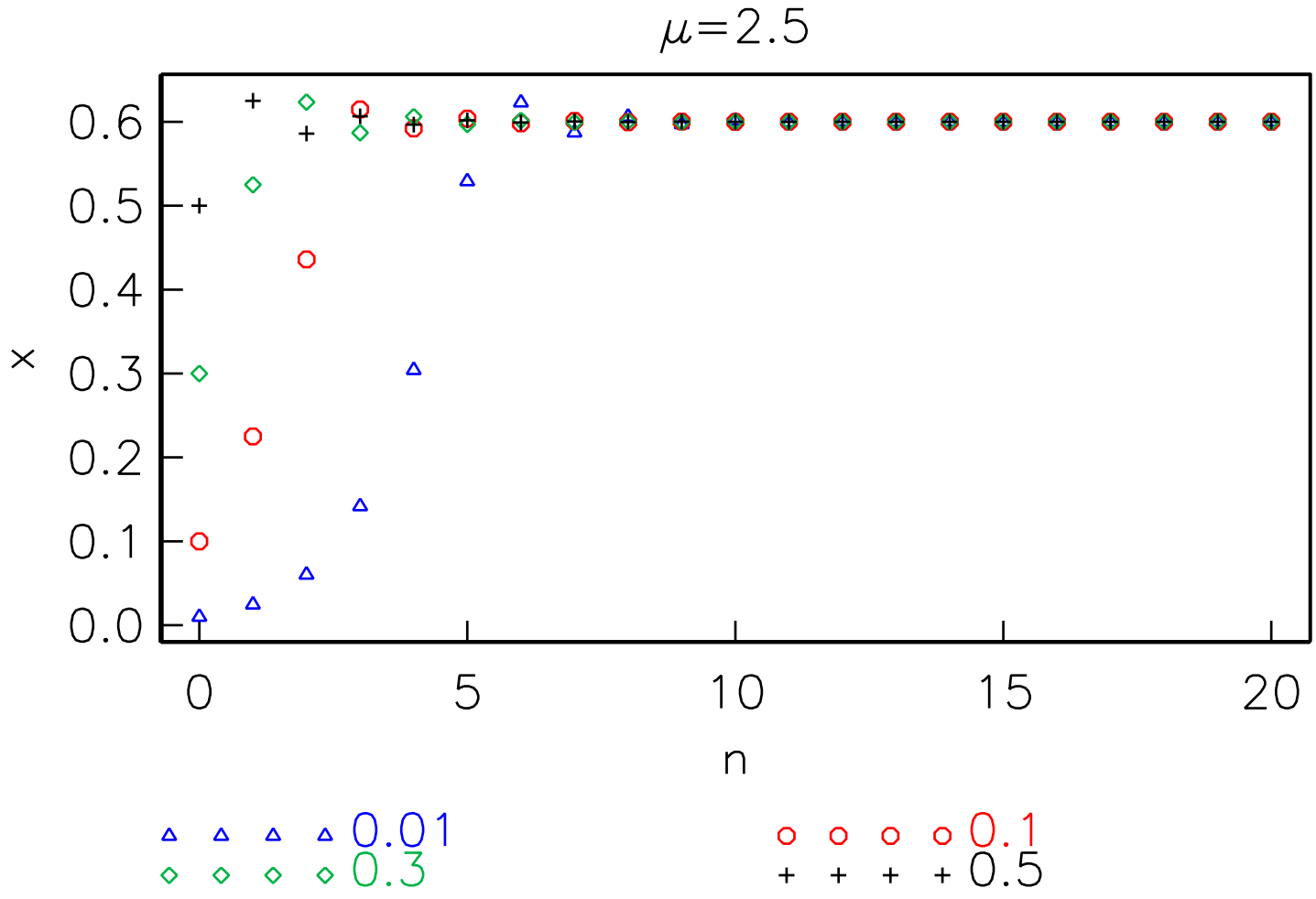
$$x_e = \mu x_e(1 - x_e)$$

som gir

$$x_e = \frac{\mu - 1}{\mu}$$

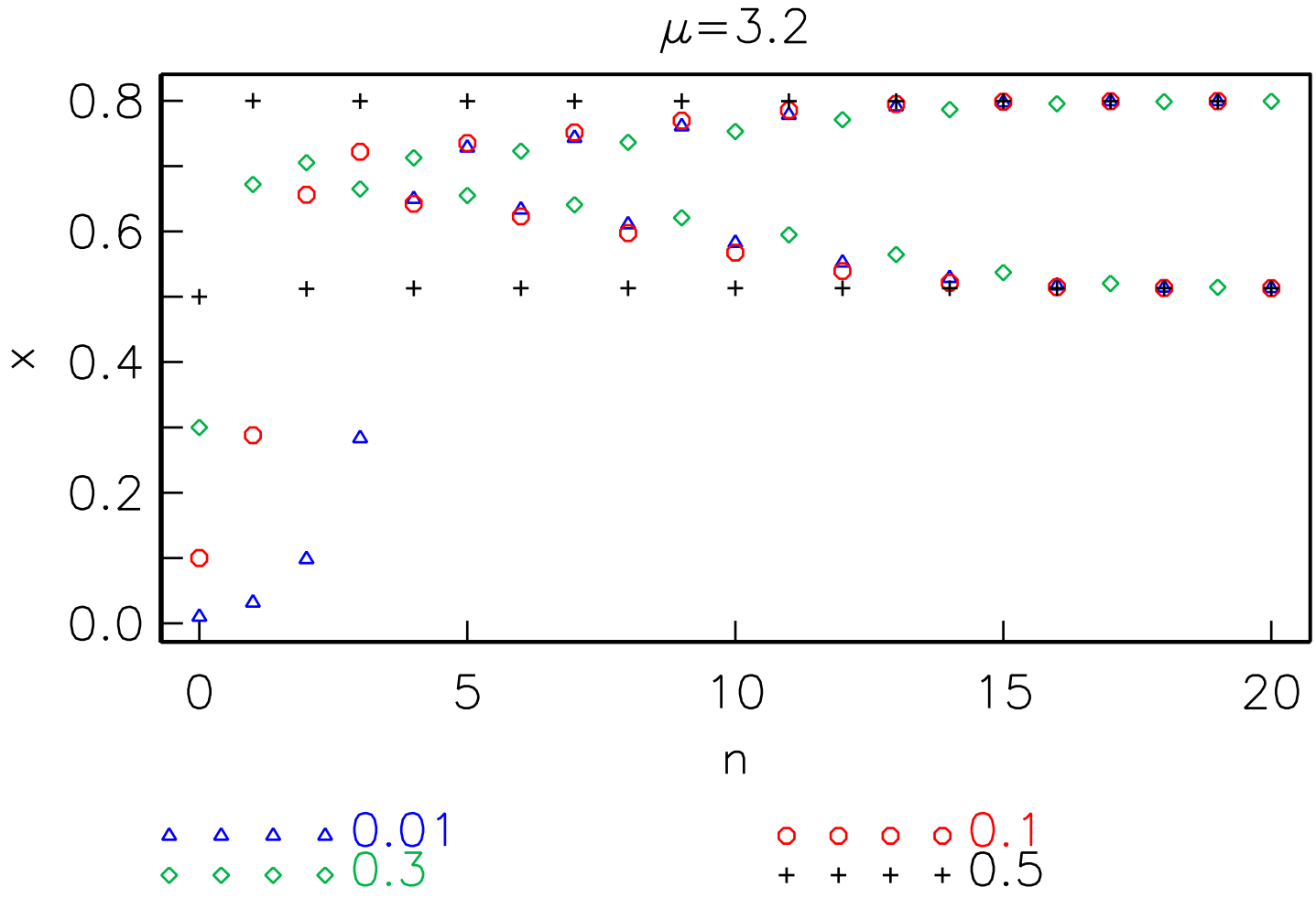
x_e er en likevektsløsning

$\mu = 2.5, \quad x_e = 0.6$



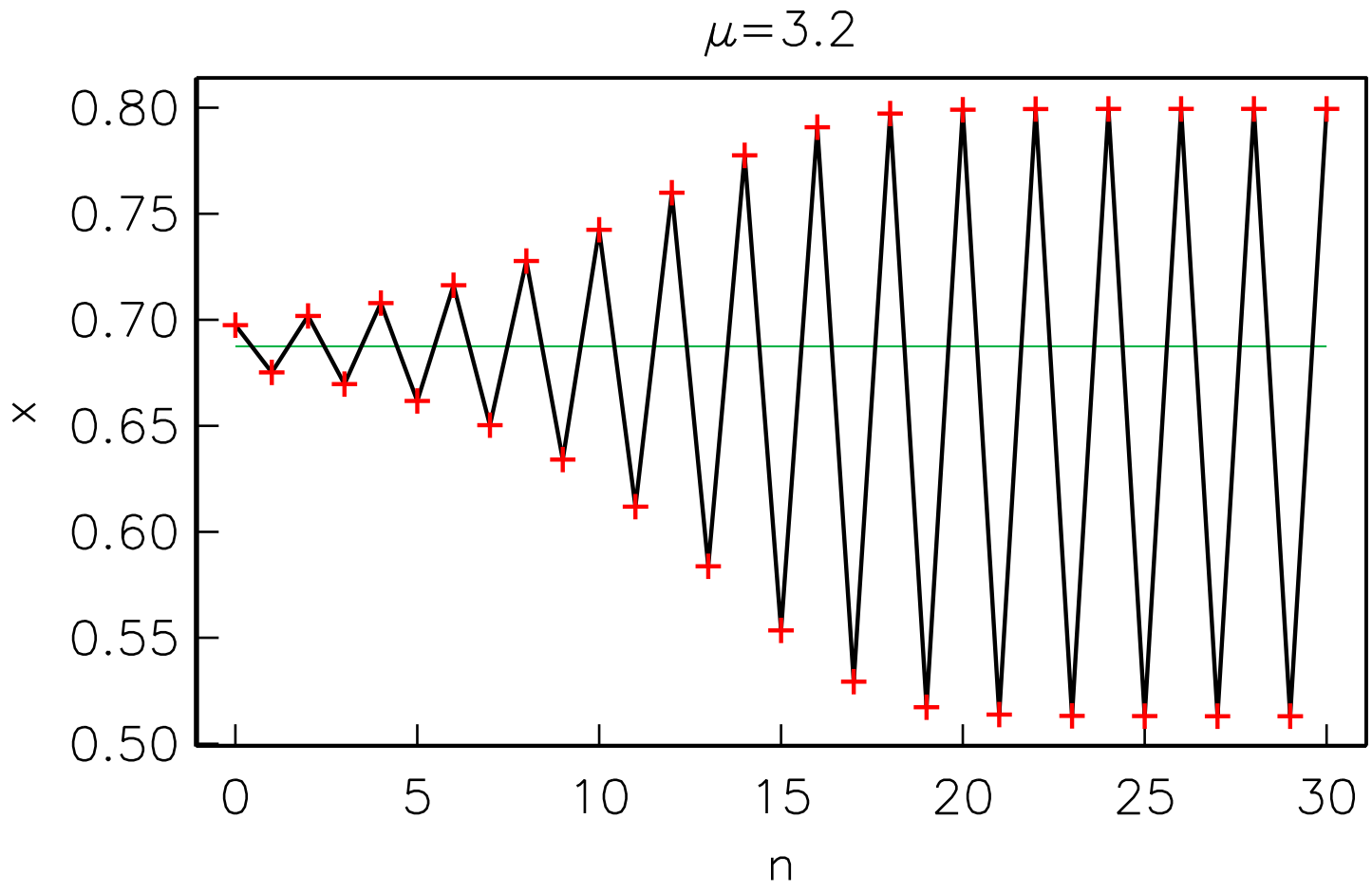
Stadig konvergens når $n \to \infty$.

$\mu = 3.2, \quad x_e = 0.69..$



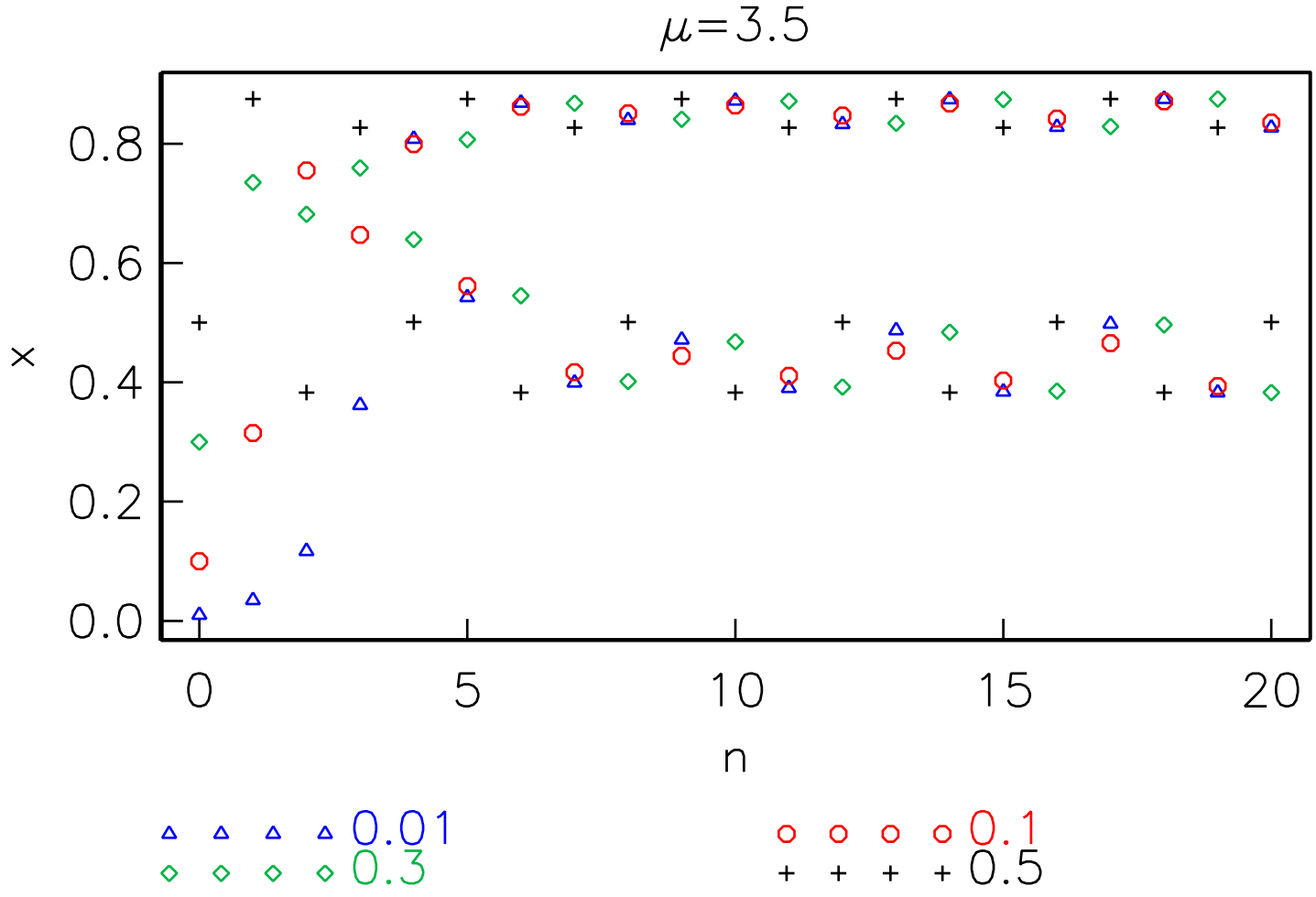
x_n nærmer seg 2-periodisk når $n \to \infty$.

$$\mu = 3.2, \quad x_0 = x_e + 0.01$$



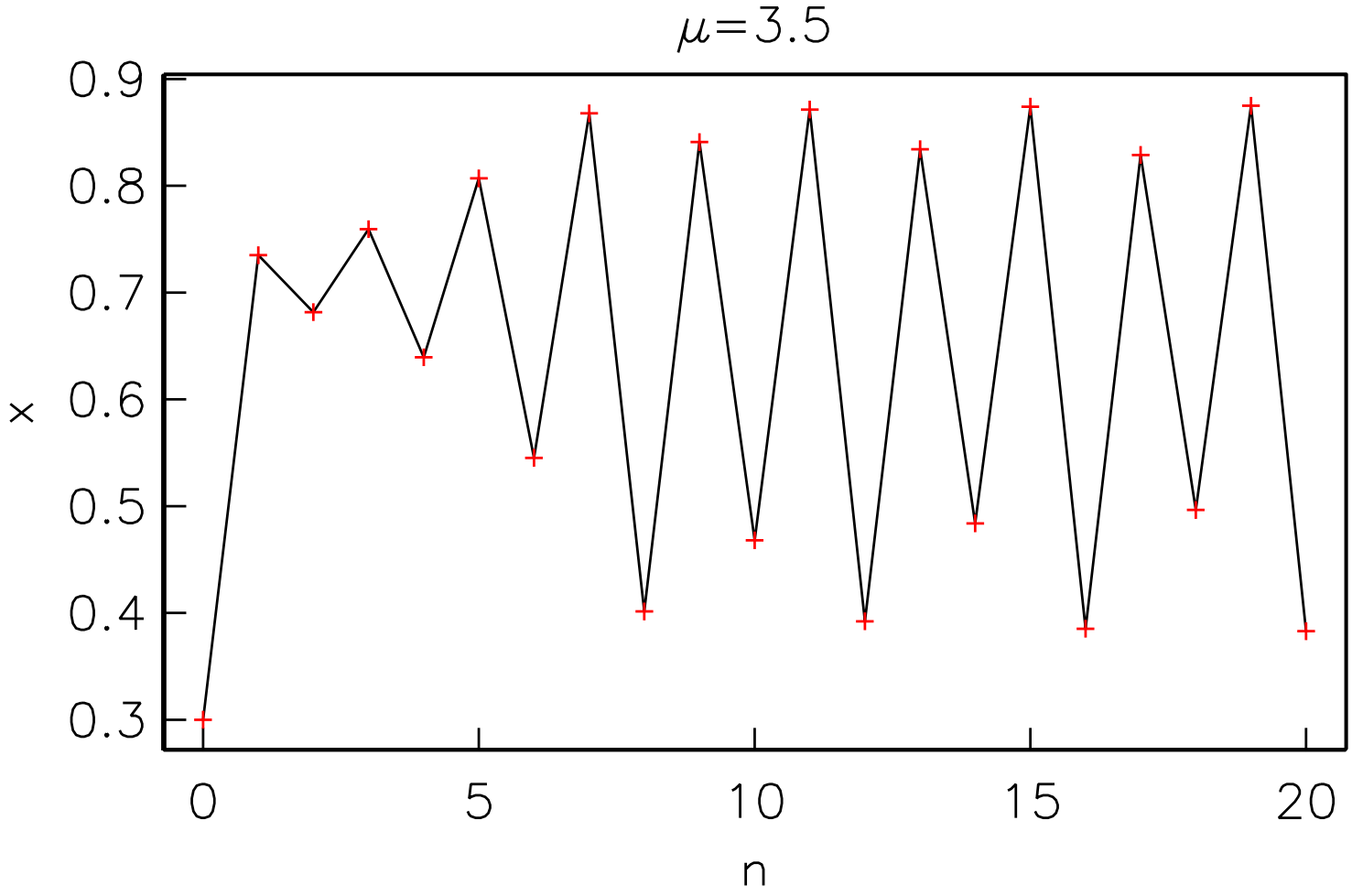
Likevekt er instabil, omstiller seg til periodisk løsning.
(Kan vise at dette gjelder for $\mu > 3$)

$\mu = 3.5, \quad x_e = 0.71..$



Større perioder når $n \to \infty$.

$\mu = 3.5$, $x_e = 0.71..$



$x_0 = 0.3$.

Bemerkninger

Simuleringer antyder

- Likevekt instiller seg for små μ
- Likevekt instabil for større $\mu \Rightarrow$ periodisk løsning instiller seg
- Mer komplekst bilde med økende μ

Kunne gjort teori for

- Stabilitet av likvekt
- Forklaring (matematisk) av de periodiske løsningen
- ...

Enkelt eksempel på utvikling mot kaos (se slutten av kap. 4 i Kalkulus)
Virkelige populasjonsvariasjoner har ofte et periodisk preg.