Differenslikninger

Ole Christian Lingjærde, Dept of Informatics, UiO

30. september 2021

Dagens agenda

- Eksempler på differenslikninger
- Hvordan løse differenslikninger i Python
- Oppgave 5.39, A.1 og A.4 fra boka til Langtangen

Differenslikninger

Differenslikninger brukes til å beskrive sammenhengen mellom ulike ledd i en matematisk følge $x_1, x_2, x_3, \ldots$. Eksempel:

$$x_n = x_{n-1} - 3\cos x_{n-2}, \qquad n = 2, 3, \ldots$$

Å løse en differenslikning betyr å finne verdiene til $x_1, x_2, x_3, \ldots$.

Initialbetingelser

For å løse en differenslikning må vi kjenne verdiene til noen av x_i'ene (typisk de første elementene i følgen). Dette kalles *initialbetingelsene*. Eksempel:

$$\begin{aligned} x_1 &= 0 \\ x_n &= 1 + 0.8x_{n-1} \qquad\qquad n = 2, 3, \ldots \end{aligned}$$

Her er $x_1 = 0$ initialbetingelsen.

Liveprogrammering: innskudd i banken

Vi setter 1000 kroner i banken. Hvis årlig rente er p prosent, kan vi beskrive beløpet i banken etter n år slik:

$$\begin{aligned} x_0 &= 1000 \\ x_n &= \left(1 + \frac{p}{100}\right) x_{n-1} \qquad\qquad n = 1, 2, \ldots \end{aligned}$$

Oppgave:

- Lag et program som regner ut verdien til innskuddet etter 1, 2, ..., 20 år. Startbeløp og rente leses fra bruker under kjøring. Svaret skrives ut som en tabell på skjermen og plottes med x_n på y-aksen og n på x-aksen.

- Prøv programmet med $x_0 = 1000$ og $p = 5$.

Liveprogrammering: populasjonsvekst

Vi ønsker å se hvordan antall reinsdyr på Hardangervidda endres over tid. Ved tid $t = 0$ antar vi at det er 10 000 dyr. Etter n år er antall reinsdyr tilnærmet lik

$$x_n = x_{n-1} + a \cdot x_{n-1}\left(1 - \frac{x_{n-1}}{m}\right)$$

hvor $a > 0$ og $m > 1$ er kjente parametre.

Oppgave:

- Lag et program som leser inn x_0, m og a fra kommandolinjen og som regner ut $x_1, x_2, \ldots, x_{200}$ og plotter x_n som funksjon av n.

- Prøv programmet med x_0=10000, m=12000, og a=0.1

Liveprogrammering: skrive ut rasjonale tall

Den såkalte *Stern-følgen* er gitt slik: $x_0 = 0$, $x_1 = 1$ og

$$x_{2n} = x_n$$
$$x_{2n+1} = x_n + x_{n+1}$$

Det kan vises at x_0/x_1, x_1/x_2, x_2/x_3, ... er en liste over alle positive rasjonale tall uten gjentakelser.

Oppgave:

- Lag et program som beregner $x_2, x_3, \ldots, x_R$ og som skriver ut de rasjonale tallene $x_0/x_1, \ldots, x_{99}/x_R$.

- Prøv programmet med R=15.

Plenumsoppgaver

Exercise 5.39: Animate the evolution of Taylor polynomials
A general series approximation (to a function) can be written as

$$S(x; M, N) = \sum_{k=M}^{N} f_k(x)\,.$$

For example, the Taylor polynomial of degree N for e^x equals $S(x; 0, N)$ with $f_k(x) = x^k/k!$. The purpose of the exercise is to make a movie of how $S(x; M, N)$ develops and improves as an approximation as we add terms in the sum. That is, the frames in the movie correspond to plots of $S(x; M, M)$, $S(x; M, M+1)$, $S(x; M, M+2)$, ..., $S(x; M, N)$.

Forrige gang gjorde vi deloppgave a)

Plenumsoppgaver

a) Make a function

```
animate_series(fk, M, N, xmin, xmax, ymin, ymax, n, exact)
```

for creating such animations. The argument `fk` holds a Python function implementing the term $f_k(x)$ in the sum, `M` and `N` are the summation limits, the next arguments are the minimum and maximum x and y values in the plot, `n` is the number of x points in the curves to be plotted, and `exact` holds the function that $S(x)$ aims at approximating.

Hint Here is some more information on how to write the `animate_series` function. The function must accumulate the $f_k(x)$ terms in a variable s, and for each k value, s is plotted against x together with a curve reflecting the exact function. Each plot must be saved in a file, say with names `tmp_0000.png`, `tmp_0001.png`, and so on (these filenames can be generated by `tmp_%04d.png`, using an appropriate counter). Use the `movie` function to combine all the plot files into a movie in a desired movie format.

In the beginning of the `animate_series` function, it is necessary to remove all old plot files of the form `tmp_*.png`. This can be done by the `glob` module and the `os.remove` function as exemplified in Sect. 5.3.4.

Plenumsoppgaver

b) Call the `animate_series` function for the Taylor series for $\sin x$, where $f_k(x) = (-1)^k x^{2k+1}/(2k+1)!$, and $x \in [0, 13\pi]$, $M = 0$, $N = 40$, $y \in [-2, 2]$.

c) Call the `animate_series` function for the Taylor series for e^{-x}, where $f_k(x) = (-x)^k/k!$, and $x \in [0, 15]$, $M = 0$, $N = 30$, $y \in [-0.5, 1.4]$.

Filename: `animate_Taylor_series`.

Plenumsoppgaver

Exercise A.1: Determine the limit of a sequence

a) Write a Python function for computing and returning the sequence

$$a_n = \frac{7 + 1/(n+1)}{3 - 1/(n+1)^2}, \quad n = 0, 2, \ldots, N\,.$$

Write out the sequence for $N = 100$. Find the exact limit as $N \to \infty$ and compare with a_N.

b) Write a Python function for computing and returning the sequence

$$D_n = \frac{\sin(2^{-n})}{2^{-n}}, \quad n = 0, \ldots, N\,.$$

Determine the limit of this sequence for large N.

Plenumsoppgaver

c) Given the sequence

$$D_n = \frac{f(x+h) - f(x)}{h}, \quad h = 2^{-n}, \tag{A.47}$$

make a function `D(f, x, N)` that takes a function $f(x)$, a value x, and the number N of terms in the sequence as arguments, and returns the sequence D_n for $n = 0, 1, \ldots, N$. Make a call to the `D` function with $f(x) = \sin x$, $x = 0$, and $N = 80$. Plot the evolution of the computed D_n values, using small circles for the data points.

d) Make another call to `D` where $x = \pi$ and plot this sequence in a separate figure. What would be your expected limit?

e) Explain why the computations for $x = \pi$ go wrong for large N.

Hint Print out the numerator and denominator in D_n.

Filename: `sequence_limits`.

Plenumsoppgaver

Exercise A.4: Compute the development of a loan
Solve (A.16)–(A.17) in a Python function.
Filename: `loan`.

$$y_n = \frac{p}{12 \cdot 100} x_{n-1} + \frac{L}{N} \quad \text{(A.16)}$$

$$x_n = x_{n-1} + \frac{p}{12 \cdot 100} x_{n-1} - y_n \quad \text{(A.17)}$$