Anders Johansson

Ekvationen $x^y = y^x$

Exempel på problemlösning med hjälp av programmering

Ekvationen $x^y = y^x$ kan studeras med hjälp av algebra, numerisk analys och programmering. Författaren demonstrerar bland annat de klassiska numeriska metoderna interpolation, ekvationslösning och kurvanpassning. Länkar ges till tio programmeringslösningar i Python 3 via webbplatsen repl.it.

En matematiklärarkollega hade tillsammans med sin klass noterat att talet 16 är lite speciellt då det kan skrivas både som 2^4 och 4^2. En naturlig följdfråga var då: finns det fler positiva heltalslösningar till ekvationen $x^y = y^x$? Problemet har studerats tidigare, bl a av Daniel Bernoulli, Christian Goldbach och Leonhard Euler på 1700-talet, vilket Marta Sved beskriver i en artikel i *Mathematics Magazine*. Som senare kommer att visas, kan en generell analytisk lösning tas fram, men eftersom ekvationen består av en kombination av exponential- och potensekvation, är det inte möjligt att finna en analytisk lösning x för givet y. Däremot kan numeriska metoder och programmering med fördel användas.

I ämnesplanen för matematik som träder i kraft hösten 2018, finns nya formuleringar rörande programmering som en strategi för problemlösning. Under Problemlösning i Centralt innehåll står det bl a: ”Strategier för matematisk problemlösning inklusive modellering av olika situationer, såväl med som utan digitala verktyg och programmering.” Detta gäller kurserna Matematik 1c, Matematik 2c, Matematik 3bc, Matematik 4, Matematik 5 och Matematik – specialisering. I Skolverkets kommentarmaterial kan läsas:

> *I matematik 3b samt hela c-spåret finns programmering med som en strategi för problemlösning. Formuleringen som används är medvetet öppen för att tillåta stor variation både i vilken utsträckning programmering förekommer i undervisning och i vilka former. […] Det ställs inga krav på specifika programmeringsspråk eller -miljöer. Det är dock ett krav att programmeringen används som en strategi för problemlösning.*
>
> (Om ämnet matematik)

Undersökningen av ekvationen $x^y = y^x$ erbjuder flera möjligheter till programmeringslösningar lämpliga för de nämnda kurserna. En plattformsoberoende onlineprogrammeringsmiljö är repl.it, som används i denna artikel.

Lösningar

Heltalslösningar

Trivial lösning
Den triviala lösningen till ekvationen är $x=y=a$, där a är ett godtyckligt (hel) tal. Exempelvis är några olika lösningar: $x=y=1$, $x=y=2$, $x=y=3$, ..., $x=y=a$.

Icke-triviala lösningar
För att hitta fler icke-triviala heltalslösningar än $x=2$, $y=4$, kan systematisk prövning användas, dvs för $x=1$ pröva om $y=2, 3, 4, 5$ osv ger samma resultat för Vänster Led (VL) x^y och Höger Led (HL) y^x. Fortsätt sedan med $x=2$ och $y=1, 3, 4, 5$ osv. Det är emellertid en tidsödande process att göra detta för hand.

Reella lösningar

För att ta reda på reella lösningar till ekvationen, kan en parametrisering göras där y skrivs som en reell konstant multiplicerad med x, dvs $y=kx$. Detta görs som nedan. Ursprungligen togs denna allmänna lösning fram av Goldbach, se exempelvis Sved.

$$x^y = y^x \tag{1}$$

$$x^{kx} = (kx)^x \tag{2}$$

$$(x^k)^x = (kx)^x \tag{3}$$

Eftersom exponenten är lika i VL och HL, måste även basen vara lika, varför:

$$x^k = kx \tag{4}$$

$$x^k - kx = 0 \tag{5}$$

$$x(x^{k-1} - k) = 0 \tag{6}$$

Nollproduktmetoden ger nu två nya ekvationer: $x=0$ och $x^{k-1}-k=0$. Den andra ekvationen ger vidare:

$$x^{k-1} = k \tag{7}$$

$$x = \sqrt[k-1]{k} = k^{\frac{1}{k-1}} \tag{8}$$

Detta ger nu:

$$y = kx = kk^{\frac{1}{k-1}} = k^{\frac{1}{k-1}+1} = k^{\frac{1}{k-1}+\frac{k-1}{k-1}} = k^{\frac{1+k-1}{k-1}} = k^{\frac{k}{k-1}} \tag{9}$$

$$\begin{cases} x &= k^{\frac{1}{k-1}} \\ y &= k^{\frac{k}{k-1}} \end{cases} \tag{10}$$

Då k väljs till positiva heltal, fås ekvationslösningarna i tabell 1.

k	x	y
2	2	4
3	$3^{1/2} = \sqrt{3}$	$3^{3/2} = 3\sqrt{3}$
4	$4^{1/3} = \sqrt[3]{4}$	$4^{4/3} = (\sqrt[3]{4})^4$
5	$\sqrt[4]{5}$	$(\sqrt[4]{5})^5$
6	$\sqrt[5]{6}$	$(\sqrt[5]{6})^6$
...	...	...
n	$\sqrt[n-1]{n}$	$(\sqrt[n-1]{n})^n$

Tabell 1: Reella ekvationslösningar då k antar positiva heltalsvärden större än k = 1.

Rationella lösningar

För att ta fram rationella lösningar görs, enligt Sved, omskrivningen:

$$\frac{1}{k-1} = u \text{ vilket ger } k = 1. + \frac{1}{u}$$

Detta insatt i (10) ger sedan:

$$\begin{cases} x &= \left(1+\frac{1}{u}\right)^u \\ y &= \left(1+\frac{1}{u}\right)^{u\left(1+\frac{1}{u}\right)} = \left(1+\frac{1}{u}\right)^{u+1} \end{cases} \qquad (11)$$

Då u väljs till positiva heltal, fås ekvationslösningarna i tabell 2. Sved visar även att denna sekvens av lösningar är samtliga icke-triviala rationella lösningar till ekvationen $x^y = y^x$.

u	x	y
1	$\left(\frac{2}{1}\right)^1 = 2$	$\left(\frac{2}{1}\right)^2 = 4$
2	$\left(\frac{3}{2}\right)^2 = \frac{9}{4}$	$\left(\frac{3}{2}\right)^3 = \frac{27}{8}$
3	$\left(\frac{4}{3}\right)^3 = \frac{64}{27}$	$\left(\frac{4}{3}\right)^4 = \frac{256}{81}$
4	$\left(\frac{5}{4}\right)^4 = \frac{625}{256}$	$\left(\frac{5}{4}\right)^5 = \frac{3125}{1024}$
...	...	...
n	$\left(\frac{n+1}{n}\right)^n$	$\left(\frac{n+1}{n}\right)^{n+1}$

Tabell 2: Rationella ekvationslösningar då u antar positiva heltalsvärden.

En speciell lösning

För stora värden på u, kommer både x och y att närma sig talet e, vilket t ex kan inses genom att successivt sätta in allt större tal u i ekvation (11).

$$\begin{cases} x &= \lim\limits_{u\to\infty}\left(1+\dfrac{1}{u}\right)^{u} = e = 2.71828... \\ y &= \lim\limits_{u\to\infty}\left(1+\dfrac{1}{u}\right)^{u+1} = e = 2.71828... \end{cases} \tag{12}$$

Programmering

Skolverket höll under höstterminen 2017 ett antal konferenser om programmering på olika platser runt om i Sverige. På Skolverkets webbplats återfinns material från konferenserna under rubriken *Konferenser om skolans digitalisering*. I konferensmaterialet finns både en introduktion till Python 3 och ett antal exempeluppgifter med anknytning till de olika gymnasiekurserna Ma 1 – Ma 5. Programmeringsmiljön repl.it presenterades och användes på konferenserna. Malmö stads Gymnasie- och vuxenutbildningsförvaltning erbjuder en Massive Open Online Course, MOOC, *Matematisk programmering i Python*. På Skolverkets lärportal (larportalen.skolverket.se) finns under Digital kompetens två moduler som behandlar ”Matematikundervisning med digitala verktyg” på gymnasienivå, varav den andra modulen behandlar programmering.

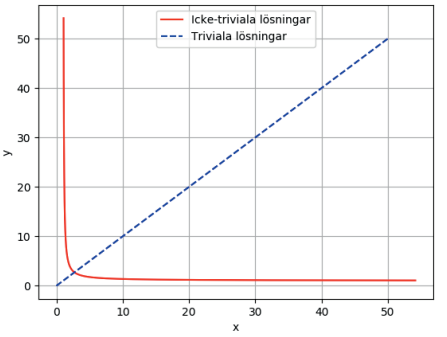
Heltalslösningar

Systematisk prövning kan utföras på ett smidigt sätt med hjälp av programmering och for-loopar. Två for-loopar nästlas, dvs skrivs inuti varandra. I den yttre loopen ökas heltalet x med ett för varje varv i loopen, $x = 1, 2, 3, 4, ...$ I den inre for-loopen ökas istället heltalet y med ett steg per varv, $y = 1, 2, 3, 4, ...$ För varje varv i den inre loopen kontrolleras om $x^y = y^x$. Detta gör att för varje x-värde prövas samtliga y-värden upp till det valda största x-värdet. Ett programförslag finns på länken repl.it/MFqf/87. Vid närmare eftertanke, behöver dock inte lika många y-värden som x-värden testas. På grund av symmetrin räcker det att för varje x-värde testa y upp till $x - 1$ i den inre for-loopen. Ett programförslag för detta: repl.it/MfTn/27.

Reella lösningar

Visualisering av lösningarna

Lösningarna kan visualiseras genom att använda ekvation (10). Ett program för att visa lösningarna: repl.it/MZwX/112. Programmet kan emellertid göras effektivare genom att skicka med arrayer direkt till funktionerna istället för att använda loopar, se: repl.it/NAlU/2.



Figur 1: Lösningar till ekvationen $x^y = y^x$. Triviala lösningar avser $x=y=1, x=y=2, x=y=e, x=y=3, x=y=\pi, x=y=3{,}465\ldots$, osv.

Från figuren kan ett antal observationer göras:

- Lösningarna representeras av två olika ”grenar”: de triviala lösningarna ligger på linjen $y=x$ och övriga lösningar ligger på den andra grenen.
- Den icke-triviala lösningsgrenen är spegelsymmetrisk och kan speglas i linjen $y=x$.
- Den enda skärningspunkten mellan linjerna är lösningen $x=y=e=2{,}71828\ldots$, dvs punkten $(e; e)=(2{,}718\ldots; 2{,}718\ldots)$.

En annan problemställning är att för ett givet y-värde bestämma x-värdet som satisfierar ekvationen. Ekvationen skrivs då om enligt följande:

$$f(x, y) = x^y - y^x = 0 \tag{13}$$

y antas vara känt, $y=b$, vilket ger:

$$f(x) = x^b - b^x = 0 \tag{14}$$

Denna ekvation kan lösas på olika sätt. Ett par metoder är intervallhalvering och Newton-Raphsons metod, vilka presenteras nedan. För mer detaljerade beskrivningar av dessa båda och även fler metoder, se exempelvis *Numerisk analys – en introduktion*.

Ekvationslösning med intervallhalvering

Intervallhalvering är en enkel och robust metod att lösa ekvationer utan tillgång till analytiska derivator. Funktionsvärden beräknas i intervallets vänstra och högra ändpunkter, $f_V = f(x_V)$, $f_H = f(x_H)$ och dess mittpunkt $f_{MITT} = f\left(\frac{x_V + x_H}{2}\right)$. Om f_V och f_{MITT} har samma tecken, befinner sig lösningen mellan $\frac{x_V + x_H}{2}$ och x_H. Intervallets nedre gräns ändras då till mittpunkten $x_V = \frac{x_V + x_H}{2}$ och detta intervall undersöks på samma sätt.

Om däremot f_V och f_{MITT} har olika tecken, ligger lösningen mellan x_V och x_{MITT} och den högra intervallgränsen flyttas till mitten,

$$x_H = \frac{x_V + x_H}{2} .$$

Iterationen fortsätter tills intervallets storlek ryms inom en given tolerans $|x_V - x_H| < tol$. En implementering i Python 3 finns på: repl.it/MY3p/256.

Ekvationslösning med Newton-Raphsons metod

Newton-Raphsons metod kan beskrivas enligt följande: Starta i punkten $(x_0, f(x_0))$ och ställ upp tangentens ekvation i denna punkt. Enligt räta linjens ekvation (enpunktsformen) ges den av:

$$y - f(x_0) = f'(x_0)(x - x_0) \tag{15}$$

Följ nu denna linje tills den skär x-axeln, vilket är i punkten $(x_1, 0)$. Sätt in denna punkt som (x, y) i (15):

$$0 - f(x_1) = f'(x_0)(x_1 - x_0) \tag{16}$$

Lös därefter ut x_1:

$$x_1 = x_0 - \frac{f(x_0)}{f'(x_0)} \tag{17}$$

Upprepa sedan proceduren för att komma fram till roten

$$x_2 = x_1 - \frac{f(x_1)}{f'(x_1)} \tag{18}$$

...

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{19}$$

För vårt exempel ges ekvationen av:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^b - b^{x_n}}{bx_n^{b-1} - b^{x_n}\ln(b)} \tag{20}$$

Om startgissningen på x_0 inte ligger alltför långt från lösningen, konvergerar Newton-Raphsons metod snabbt mot den sökta lösningen. En länk till Python 3-kod: repl.it/M395/216.

Linjär interpolation

Genom att använda ekvation (10) kan lösningar (x, y) för ett antal olika värden på k erhållas, se exempelvis tabell 1 och figur 1. Om ett mellanliggande x-värde däremot är givet och motsvarande y-värde söks, kan linjär interpolation användas. Då letas de båda x-värden som ligger närmast på ömse sidor om det sökta x-värdet fram och räta linjens ekvation används för att få fram ett motsvarande värde på y. En länk till Python 3-kod för linjär interpolation: repl.it/NNxl/2.

Kurvanpassning till en enklare ekvation
Studier av funktionen för de icke-triviala lösningarna i figur 1, visar att den aktuella kurvan:

- är spegelsymmetrisk kring $y = x$
- har en horisontell asymptot $y = 1$
- har en vertikal asymptot $x = 1$.

En funktion med horisontell och vertikal asymptot är: $y(x) = \dfrac{a}{x-b} + c$.

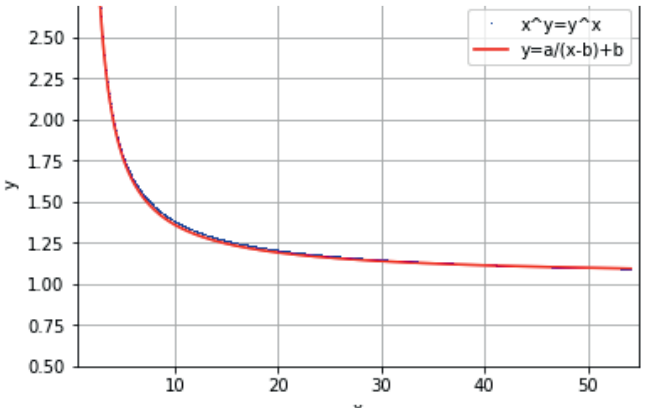
Detta ger den inversa funktionen $x(y) = \dfrac{a}{y-c} + b$.

För att få symmetriegenskapen, så krävs att $b = c$, vilket ger följande funktioner:

$$y(x) = \frac{a}{x-b} + b \tag{21}$$

$$x(y) = \frac{a}{y-b} + b \tag{22}$$

För att bestämma parametrarna a och b görs en icke-linjär kurvanpassning med t ex Levenberg-Marquardts metod. Ett exempel på en kurvanpassning ger värdena $a = 2{,}87$ och $b = 1{,}04$ och redovisas i figur 2. För mer information, se exempelvis *Linjär och ickelinjär optimering* samt följande länk till Python 3-kod: repl.it/NCAR/3.



Figur 2: Jämförelse mellan kurvanpassning och lösningar från ekvation (10).

Jämförelse mellan olika sätt att bestämma reella lösningar
En jämförelse mellan olika lösningsmetoder ger en god överensstämmelse mellan interpolationsmetoden

och Newton-Raphsons metod. Den förenklade ekvationen $y = \dfrac{a}{x-b} + b$

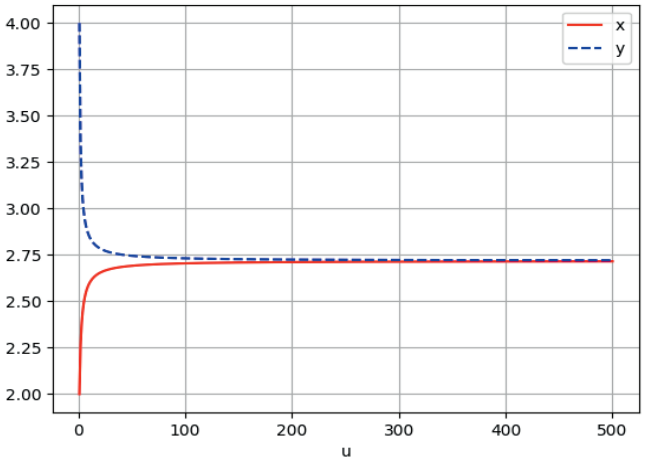
ger en sämre överensstämmelse med avvikelser på cirka 1 %. För $x = 10$ fås exempelvis lösningen $y = 1{,}371$ både för interpolation och Newton-Raphsons metod, men värdet $y = 1{,}360$ för den förenklade ekvationen.

Rationella lösningar

Ett program som skriver ut lösningarna presenterade i tabell 2, baserat på ekvation (11) finns på: repl.it/MfVC/26.

En speciell lösning

I figur 3 redovisas hur x respektive y beräknade med ekvation (12) närmar sig talet e, då u ökas. Notera att x konvergerar nedifrån och y uppifrån. Ett Python 3-program för att beräkna gränsvärdena presenterade i ekvation (12): repl.it/Mzx8/64.



Figur 3: Illustration av beräkning av gränsvärdet presenterade i ekvation (12).

Programmeringslösningar i Python 3

Följande sammanställning finns även klickbar på Nämnaren på nätet.

Systematisk prövning:
– Grundversion: repl.it/MFqf/87
– Förbättrad version: repl.it/MfTn/27

Visualisering av lösningar:
– Grundversion: repl.it/MZwX/112
– Förbättrad version: repl.it/NAlU/2

Intervallhalvering: repl.it/MY3p/256

Newton-Raphsons metod: repl.it/M395/216

Linjär interpolation: repl.it/NNxl/2

Kurvanpassning: repl.it/NCAR/3

Rationella lösningar: repl.it/MfVC/26

Gränsvärden: repl.it/Mzx8/64

Användning i undervisningen på gymnasiet

Numeriska metoder och programmeringslösningar presenterade i denna artikel kan användas i undervisningen på olika sätt.

- Den systematiska prövningen med två nästlade loopar för att finna heltalslösningar kan t ex användas i Ma 1c.
- Ekvationslösning med intervallhalvering kan exempelvis användas i Ma 2c eller Ma 3bc då eleverna känner sig vana vid algebraisk ekvationslösning.
- Implementeringen av den allmänna lösningen med de parametriserade ekvationerna $x = k^{\frac{1}{k-1}}$, $y = k^{\frac{k}{k-1}}$ används förslagsvis först i Ma 2c, då förståelse för bland annat nollproduktmetoden krävs för att genomföra och förstå den generella lösningen.
- Linjär interpolation bygger på räta linjens ekvation och kan användas i Ma 2c.
- Beräkningen av gränsvärdena:

$$x = \lim_{u\to\infty}\left(1+\frac{1}{u}\right)^{u},\ y = \lim_{u\to\infty}\left(1+\tfrac{1}{u}\right)^{u+1}$$

  ger förhoppningsvis eleverna lite bättre förståelse för gränsvärden i Ma 3bc.
- Ekvationslösning med Newton-Raphsons metod och kurvanpassning är lite mer avancerade numeriska metoder, vilka förslagsvis passar bäst i de högre matematikkurserna Ma 4, Ma 5 eller Ma – specialisering.

LITTERATUR

Dahlström, M., Enving, S., Strid, K., Zanjani, N., Friberg, C. & Sandgren, M. (2017). Matematisk programmering i Python, sites.google.com/skola.malmo.se/programmeringsfortbildning/startsida

Sved, M. (1990). On the rational solutions of $x^y = y^x$. *Mathematics Magazine*, 63(1), 30–33.

Eldén, L. & Wittmeyer-Koch, L. (1996). *Numerisk analys – en introduktion*. Lund: Studentlitteratur.

Lundgren, J., Rönnqvist, M. & Värbrand, P. (2001). *Linjär och ickelinjär optimering*. Lund: Studentlitteratur.

På Nämnaren på nätet finner du klickbara länkar.