Ordinära differentialekvationer 1

I en serie studioövningar ska vi lära oss hur man konstruerar och beräknar lösningar till ordinära differentialekvationer av typen

$$u'(x) = f(x, u(x))$$

Vi börjar i denna övning med det enklaste fallet då $f = f(x)$ är en funktion som bara beror på x. I senare övningar låter vi f bero också på u och till sist låter vi u och f vara vektorfunktioner, dvs vi löser system av ordinära differentialekvationer.

1.1 Primitiv funktion

Antag att $f : I \to \mathbf{R}$ är en funktion. Om funktionen u är deriverbar och uppfyller

$$u'(x) = f(x), \quad \forall x \in I, \tag{1}$$

så säger vi att u är en *primitiv funktion* till f. Vi ska konstruera primitiva funktioner. Vi börjar med att notera att primitiv funktion är väsentligen unik. För om två funktioner u och v är primitiva funktioner till samma funktion f så gäller

$$(u - v)'(x) = u'(x) - v'(x) = f(x) - f(x) = 0, \quad \forall x \in I.$$

Härav följer att

$$u - v = \text{ konstant}$$

dvs

$$u(x) = v(x) + C.$$

För att bestämma konstanten kan man kräva att funktionen skall vara lika med ett bestämt värde i en bestämd punkt: $u(a) = u_a$. Vi kan nu precisera vår uppgift. Givet en kontinuerlig funktion $f : [a, b] \to \mathbf{R}$ och en konstant u_a söker vi en deriverbar funktion u sådan att

$$\begin{aligned} &u'(x) = f(x), \quad x \in [a, b], \\ &u(a) = u_a. \end{aligned} \tag{2}$$

Eftersom vi anger funktionens värde i intervallets vänstra ändpunkt så kallas problemet (2) ett *begynnelsevärdesproblem*.

Exempel 1. Om man känner ett par av en funktion och dess derivata så kan man lösa motsvarande begynnelsevärdesproblem. Till exempel känner vi till följande derivata:

$$Dx^m = mx^{m-1} \quad (m \geq 0)$$

Detta leder till

$$D\frac{x^m}{m} = x^{m-1} \quad (m > 0)$$

dvs, med $r = m - 1$,

$$D\frac{x^{r+1}}{r+1} = x^r \quad (r > -1)$$

Alltså kan vi säga att problemet

$$u'(x) = x^r \quad (r > -1)$$

har lösningen

$$u(x) = \frac{x^{r+1}}{r+1} + C.$$

Vi bestämmer konstanten, till exemel, genom begynnelsevillkoret

$$u(0) = 0$$

vilket leder till $C = 0$ och $u(x) = \frac{x^{r+1}}{r+1}$. Mer allmänt: villkoret $u(0) = u_0$ leder till $C = u_0$ och $u(x) = \frac{x^{r+1}}{r+1} + u_0$. Alltså har begynnelsevärdesproblemet

$$\begin{aligned} u'(x) &= x^r, \quad x \in [0, \infty), \\ u(0) &= u_0, \end{aligned}$$

den unika lösningen

$$u(x) = \frac{x^{r+1}}{r+1} + u_0.$$

□

Denna metod fungerar bara om vi redan känner ett funktion/derivata-par. Vi ska nu beskriva hur man kan konstruera en primitiv funktion för varje kontinuerlig funktion f, dvs konstruera en unik lösning till begynnelsevärdesproblemet (2). Vi börjar med att dela in intervallet $[a, b]$ i N stycken delintervall av längden $h = (b - a)/N$:

$$\begin{aligned} &a = x_0 < x_1 < x_2 < \dots x_{i-1} < x_i < \cdots < x_{N-1} < x_N = b, \\ &x_i = a + hi, \ h = (b - a)/N = x_i - x_{i-1}. \end{aligned}$$

Sedan utgår vi från approximationen

$$\frac{u(x_i) - u(x_{i-1})}{h} \approx u'(x_{i-1}) = f(x_{i-1})$$

vilket leder till

$$u(x_i) \approx u(x_{i-1}) + hf(x_{i-1}).$$

Vi beräknar nu en approximativ lösning enligt

$$\begin{aligned} U(x_0) &= u_a \\ U(x_i) &= U(x_{i-1}) + hf(x_{i-1}). \end{aligned}$$

Genom att förbinda punkterna $(x_i, U(x_i))$ med räta linjer får vi en graf och funktionen $U(x)$ blir definierad också mellan beräkningsnoderna x_i. Rita figur! Denna algoritm kallas Eulers metod.

Konvergens

Man kan visa att $U(x)$ konvergerar mot en unik lösning $u(x)$ till (2) då antalet delintervall $N \to \infty$. Beviset är upplagt enligt samma princip som för bisektionsalgoritmen och fixpunktsiterationen.

Vi delar intervallet $[a, b]$ med hjälp av halvering. Vi låter n beteckna antalet halveringar, antalet delintervall blir $N = 2^n$, och för varje n får vi en approximativ lösning $U_n(x)$:

$$\begin{aligned} &n = 1, \quad N = 2^1 = 2, \quad U_1(x) \\ &n = 2, \quad N = 2^2 = 4, \quad U_2(x) \\ &n = 3, \quad N = 2^3 = 8, \quad U_3(x) \\ &n = 4, \quad N = 2^4 = 16, \quad U_4(x) \\ &\text{och så vidare.} \end{aligned}$$

Beviset går sedan enligt följande.

1. Algoritmen Eulers metod ger en följd av approximativa lösningar $\{U_n(x)\}_{n=1}^{\infty}$.
2. Vi visar att $U_n(x)$ är en Cauchy-följd, dvs $U_n(x) - U_m(x) \to 0$ då $m, n \to \infty$. Vi får alltså ett reellt tal (decimalutveckling) $u(x) = \lim_{n\infty} U_n(x)$.
3. Vi visar att u löser (2), dvs u är deriverbar med $u' = f$ och $u(a) = u_a$.
4. Vi visar att lösningen till (2) är unik. (Det har vi redan gjort ovan.)

Detta bevis är ganska långt och lite för svårt och vi nöjer oss med att nämna dessa steg.

Implementering i MATLAB

Detta är lätt programmera i MATLAB. Algoritmen är följande. Eftersom vi behöver både x_i och $U(x_i)$ för att till exempel plotta funktionen så måste algoritmen räkna ut även noderna x_i.

$$\text{initiera:} \begin{cases} x_0 = a \\ U(x_0) = u_a \end{cases}$$

$$\text{uppdatera:} \begin{cases} \text{while } x_i < b \\ x_i = x_{i-1} + h \\ U(x_i) = U(x_{i-1}) + hf(x_{i-1}). \end{cases}$$

I MATLAB kan index inte börja med 0, så att till exempel initieringen av x skrivs `x(1)=a`. Likaså måste $U(x_i)$ representeras av en vektor `U` med elementen `U(i)`. Man kan inte skriva `U(x(i))`.

Övning 1. Skriv ett program `myprim.m` med anropet `[x,U]=myprim(f,I,ua,h)` som löser begynnelsevärdesproblemet (2). Du skall använda programskalet `myprim.m`. In- och ut-variablerna förklaras i programskalet.

Testa programmet på följande. Lös först begynnelsevärdesproblemet analytiskt (dvs som en formel med penna och papper). Plotta både den analytiska lösningen u och den approximativa lösningen U i samma figur. Använd stort steg h, till exempel, $h = 10^{-1}$, när du först skriver programmet så blir det lättare att se vad som händer. Tag sedan litet h, till exempel, $h = 10^{-3}$.

$$(a) \quad \begin{aligned} &u'(x) = x^2, \quad x \in [0, 2], \\ &u(0) = 3, \end{aligned}$$

$$(b) \quad \begin{aligned} &u'(x) = x^2, \quad x \in [1, 4], \\ &u(1) = 3, \end{aligned}$$

$$(c) \quad \begin{aligned} &u'(x) = \cos(2x), \quad x \in [0, \pi], \\ &u(0) = 0, \end{aligned}$$

$$(d)\quad \begin{aligned} &u'(x) = 1/x, \quad x \in [1, 10], \\ &u(1) = 0, \end{aligned}$$

Obs att naturliga logarimen $\ln(x)$ heter `log(x)` i MATLAB. □

Vi får en noggrannare approximation om vi utgår från

$$\frac{u(x_i) - u(x_{i-1})}{h} \approx u'(\hat{x}_i) = f(\hat{x}_i), \quad \hat{x}_i = (x_i + x_{i-1})/2,$$

vilket leder till

$$u(x_i) \approx u(x_{i-1}) + hf(\hat{x}_{i-1}).$$

Vi beräknar nu en approximativ lösning enligt

$$\begin{aligned} U(x_0) &= u_a \\ U(x_i) &= U(x_{i-1}) + hf(\hat{x}_i). \end{aligned}$$

Denna algoritm kallas mittpunktsmetoden.

Övning 2. Modifiera programmet `myprim.m` så att det använder mittpunktsmetoden istället. Prova med samma exempel som ovan.

`2006-11-14 /stig`