Numerisk beräkning av primitiv funktion[1]

1.1 Primitiv funktion

Antag att $f(x)$ är en funktion definierad för $x \in [a, b]$. Om funktionen u är deriverbar och uppfyller

$$u'(x) = f(x), \qquad \forall x \in [a, b], \tag{1}$$

så säger man att u är en primitiv funktion till f . Vi ska här numeriskt beräkna primitiva funktioner och, som bieffekt, få ett alternativt sätt (jämfört med en direkt Riemannsumma) att approximera integralen

$$\int_a^b f(x)\,dx = u(b) - u(a).$$

Som påpekas i Adams-Essex (i beviset av sats 5, sid 311) gäller att två olika primitiva funktioner bara skiljer sig åt med en konstant, dvs. om två funktioner $u(x)$ och $v(x)$ är primitiva funktioner till samma funktion $f(x)$ så följer det att

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

För att bestämma konstanten kan man kräva att funktionen skall vara lika med ett bestämt värde i en bestämd punkt: $u(a) = u_a$. Givet en kontinuerlig funktion $f(x)$ på $[a, b]$ och en konstant u_a söker vi alltså en deriverbar funktion u sådan att

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

Om man tolkar variabeln x som tid som ökar från tiden a till tiden b, ser man att funktionens värde anges vid starttiden; därför kallas problemet ett *begynnelsevärdesproblem*. Analytiska lösningar till detta problem kan man bara finna för vissa funktioner $f(x)$ (de vi övar på!). Vi ska nu beskriva hur man kan numeriskt konstruera en primitiv funktion för varje kontinuerlig funktion $f(x)$, dvs konstruera en unik lösning till begynnelsevärdesproblemet (2). Vi börjar med att dela in intervallet $[a, b]$ i N stycken delintervall av längden $h = (b - a)/N$:

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

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.

Man kan visa att $U(x)$ konvergerar mot en unik lösning $u(x)$ till (2) då antalet delintervall $N \to \infty$; detta bevis är något komplicerat och tas inte upp här.

[1] 2009–11–03, Peter Hansbo, Matematiska vetenskaper Chalmers tekniska högskola

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. (facit). In- och utvariablerna 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. □

Analogt med mittpunktsregeln för integration får man en noggrannare approximation om man 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.

`2009-11-03 /peter`