TMV035 Analysis and Linear Algebra B[1]

Ordinära differentialekvationer 4

1.1 Framlänges Euler

Vi har löst det allmänna begynnelsevärdesproblemet

$$\begin{aligned} &u'(t) = f(t, u(t)), \quad t \in [a, b], \\ &u(a) = u_a, \end{aligned} \tag{1}$$

med Eulers metod (framlänges Euler)

$$\begin{aligned} &U(t_0) = u_a \\ &U(t_i) = U(t_{i-1}) + hf(t_{i-1}, U(t_{i-1})). \end{aligned}$$

Algoritmen kallas *explicit* därför att i varje steg beräknar vi den nya kolonnvektorn $U(t_i)$ som en explicit funktion av t_{i-1} och den förra kolonnvektorn $U(t_{i-1})$.

1.2 Baklänges Euler

Om vi istället utgår från

$$\frac{u(t_i) - u(t_{i-1})}{h} \approx u'(t_i) = f(t_i, u(t_i))$$

så får vi baklänges Eulers metod:

$$\begin{aligned} &U(t_0) = u_a \\ &U(t_i) = U(t_{i-1}) + hf(t_i, U(t_i)). \end{aligned}$$

Denna metod är *implicit* därför att vi måste lösa ut den nya vektorn $U(t_i)$ ur ett ekvationssystem. Närmare bestämt är $V = U(t_i)$ lösning till fixpunktsekvationen

$$V = U(t_{i-1}) + hf(t_i, V),$$

dvs

$$V = g(V), \quad \text{med } g(V) = U(t_{i-1}) + hf(t_i, V).$$

Vi kan lösa denna med fixpunktsiterationen

$$\begin{aligned} V_0 &= U(t_{i-1}), \\ V_k &= g(V_{k-1}). \end{aligned}$$

Detta fungerar om g är en kontraktion, dvs $L_g < 1$. Vi kollar detta:

$$\begin{aligned} \|g(V) - g(W)\| &= \|U(t_{i-1}) + hf(t_i, V) - U(t_{i-1}) - hf(t_i, W)\| \\ &= \|hf(t_i, V) - hf(t_i, W)\| = h\|f(t_i, V) - f(t_i, W)\| \le hL_f\|V - W\|. \end{aligned}$$

Detta betyder att

$$L_g \le hL_f < 1,$$

dvs vi har en kontraktion, om steget väljs tillräckligt litet:

$$h < 1/L_f.$$

[1] `2006-11-19 /stig`

1.3 Mittpunktsmetoden

Om vi istället utgår från

$$\frac{u(t_i) - u(t_{i-1})}{h} \approx u'(\hat{t}_i) = f(\hat{t}_i, u(\hat{t}_i)), \quad \hat{t}_i = (t_{i-1} + t_i)/2,$$

så får vi mittpunktsmetoden

$$\begin{aligned} U(t_0) &= u_a \\ U(t_i) &= U(t_{i-1}) + hf\Big(\frac{t_{i-1}+t_i}{2}, \frac{U(t_{i-1}) + U(t_i)}{2}\Big). \end{aligned}$$

Denna metod är också implicit: den nya vektorn $U(t_i)$ fås genom att lösa fixpunktsekvationen

$$V = U(t_{i-1}) + hf\Big(\frac{t_{i-1}+t_i}{2}, \frac{U(t_{i-1}) + V}{2}\Big). \tag{2}$$

Detta fungerar om $\frac{1}{2}hL_f < 1$.

Implementering i Matlab

Algoritmen är

$$\text{initiera:} \begin{cases} t_0 = a \\ U(t_0) = u_a \end{cases}$$

$$\text{uppdatera:} \begin{cases} \text{while } t_i < b \\ t_i = t_{i-1} + h \\ \text{lös ekvation (2)} \\ U(t_i) = V \end{cases}$$

Ekvation (2) löses här med algoritmen

$$\begin{aligned} &V = U(t_{i-1}) \\ &\text{while } e > \text{tol} \\ &W = V \\ &V = U(t_{i-1}) + hf\Big(\frac{t_{i-1}+t_i}{2}, \frac{U(t_{i-1}) + V}{2}\Big) \\ &e = \|V - W\| \end{aligned}$$

Lagom värde på toleransen är h^3 eftersom felet i $U(t_i)$ är ungefär h^3 för mittpunktsmetoden.

På liknande sätt kan man implementera baklänges Euler. Lagom värde på toleransen är h^2 eftersom felet i $U(t_i)$ är ungefär h^2 för Eulers metod.

Övning 1. Skriv programmen `myode1.m` och `myode2.m` med baklänges Euler respektive mittpunktsmetoden. Se `matlab/facit` på studio-sidan.

Prova programmen på samma begynnelsevärdesproblem som i förra övningen, inkl Volterra-Lotka. Notera speciellt skillnaden i beteende mellan programmen på exempel (c). □

Matlabs egna ODE-lösare

Matlab har flera program som öser ODE med samma anrop som `myode.m` utom att man inte behöver ange steget; det väljs adaptivt av programmet. Till exempel,

```
>> [t,U]=ode23(f,I,ua)
>> [t,U]=ode45(f,I,ua)
>> [t,U]=ode15s(f,I,ua)
```

Prova dessa program på samma exempel som ovan. Notera hur steget väljs genom att bilda `h=diff(t)`.