Ordinära differentialekvationer (ODE)

- Vi studerar numerisk approximation av begyllelsevärdesproblem på formen

$$y'(t) = f(t, y(t)), \quad t > a$$
$$y(a) = y_0$$

- Funktionerna *y* och f kan vara skalära eller vektorer
- Högre order ekvationer kan skrivas om på första ordnings form

$$y''(t) = f(t, y, y'), \quad t > a$$
$$y(a) = \alpha, \quad y'(a) = \beta$$

$$z = y' \rightarrow \frac{d}{dt}\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} z \\ f(t, y, z) \end{pmatrix}, \quad \begin{pmatrix} y(a) \\ z(a) \end{pmatrix} = \begin{pmatrix} \alpha \\ \beta \end{pmatrix}$$

ODE

- Begynnelsevärdena behövs för att få entydig lösning
- Extra villkor kan också ges som randvillkor

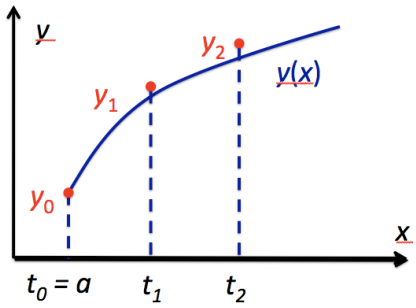
$$y''(x) = f(x, y, y'), \quad a < x < b$$
$$y(a) = \alpha, \quad y(b) = \beta$$

- Notera att en andra ornings skalär ekvation behöver två extra villkor för entydighet

Differensapproximation, begynnelsevärdesproblem

- Den obekanta funktionen $y(t)$ approximaras numeriskt för diskreta t-värden

$$y_n \approx y(t_n), \quad t_n = a + nh, \; n = 0,1,..$$



Eulers metod

- Derivatan i differentialekvtionen ersättes med dividerad differens

$$y'(t_n) \approx \frac{y(t_{n+1}) - y(t_n)}{h} \rightarrow \frac{y_{n+1} - y_n}{h}$$

- Det resulterar i Eulers metod

$$y'(t) = f(t, y(t)), \quad t > a$$
$$y(a) = \alpha$$
$$\rightarrow \begin{cases} y_0 = \alpha \\ \dfrac{y_{n+1} - y_n}{h} = f(t_n, y_n), \quad n = 0,1,.. \end{cases}$$
$$eller \quad y_{n+1} = y_n + hf(t_n, y_n), \quad n = 0,1,..$$

Numeriskt fel

- Det lokala trunkationsfelet genom att approximera derivatan ges av Taylorutveckling

$$y_{n+1} = y_n + hf(t_n, y_n) \rightarrow$$

$$y(t_{n+1}) = y(t_n) + hy'(t_n) + \frac{h^2}{2} y''(\tau) =$$

$$= y(t_n) + hf(t_n, y(t_n)) + \frac{h^2}{2} y''(\tau)$$

$$\rightarrow localt\ fel: \frac{h^2}{2} y''(\tau)$$

- Det lokala felet upprepas för alla steg i iterationen: Efter (*b*-*a*)/h steg blir totala effekten *O*(*h*). Euler är av första ordningen.

Högre ordings metoder

- En dividerad differens är den bästa approximationen av derivatan I mittpunkten mellan t_n och t_{n+1}. Det leder till trapetsmetoden for ODE

$$y_{n+1} = y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_{n+1})\right)$$

- Med Taylorutveckling kan man visa att metoden är av andra ordningen (lokalt fel $O(h^3)$, globalt fel $O(h^2)$)
- Metoden är implicit, dvs. ekvationslösning behövs i varje steg för att beräkna y_{n+1}
- Eulers metod är explicit då y_{n+1} bara finns i vänsterledet och ges av en explicit formel

Runge-Kutta metoder

- Om y_{n+1} i trapetsmetoden ersättes med sin Euler approximation erhålles en andra ordnings metod som är explicit

$$y_{n+1} = y_n + \frac{h}{2}\left(f(t_n, y_n) + f(t_{n+1}, y_n + f(t_n, y_n))\right)$$

- Vi kan formulera algoritmen ovan (RK2) på följande form som också passar för formulering av fjärde ordningens Runge-Kutta metod (RK4)

$$RK2: \quad y_{n+1} = y_n + \frac{h}{2}(f_1 + f_2)$$
$$f_1 = f(t_n, y_n)$$
$$f_2 = f(t_n + h, y_n + hf_1)$$

Runge-Kutta metoder

- Den klassiska Runge-Kutta metoden (RK4) är av fjärde ordningen, dvs. Det totala felet är av storleksordongen $O(h^4)$

$$RK4: \quad y_{n+1} = y_n + \frac{h}{6}(f_1 + 2f_2 + 2f_3 + f_4)$$
$$f_1 = f(t_n, y_n)$$
$$f_2 = f(t_n + h/2, y_n + hf_1/2)$$
$$f_2 = f(t_n + h/2, y_n + hf_2/2)$$
$$f_2 = f(t_n + h, y_n + hf_3)$$

Flerstegsmetoder

- Runge –Kutta metoder får högre ordning genom att funktionen *f* beräknas flera gånger i varje "steg" (från t_n till t_{n+1})
- Ett annat sätt är att involvera flera "steg" i samma algoritm
- Ett exempel är mittpunktsmetoden av andra ordningen

$$y_{n+2} = y_n + 2hf(t_{n+1}, y_{n+1})$$
$$y_0 = \alpha$$
$$y_1 = y_0 + hf(t_0, y_0)$$

- Notera att flerstegsmetoder behöver flera startvärden som måste beräknas utgående från y_0

Styva problem

- Vissa problem med "stora" koefficienter kan kräva speciella metoder för effektiv lösning. Exempel

$$y'(t) = 100(t^2 - y) + 2t \quad t > 0, \quad y(0) = 0$$
$$\rightarrow y(t) = t^2$$

- Eulers metod med *h* =0.1 ger otillfredsställande resultat

$$y_{n+1} = y_n + h\left(100(t_n^2 - y_n) + 2t_n\right)$$
$$y_1 = 0.000, \; y_2 = 0.120, \; y_3 = -0.640$$
$$\textit{Jämför} \;\; y(0.1) = 0.010, \; y(0.2) = 0.040, \; y(0.3) = 0.090$$

- Svårigheten är en kraftig fefortplantning från faktorn "-100h"

Styva problem

- Problem av den här typen där $\partial f / \partial y$ är mycket stort och har negativ realdel kallas styva och kräver vissa implicita metorder för att kunna approximeras effektivt. Exemplet med implicit Euler

$$y_{n+1} = y_n + hf(t_{n+1}, y_{n+1}), \ \textit{implicit Euler}$$
$$y_{n+1} = y_n + h\left(100(t_{n+1}^2 - y_{n+1}) + 2t_{n+1}\right), \ \textit{tillämpad på exemplet}$$
$$\rightarrow y_{n+1} = \frac{1}{11}\left(y_n + 10t_{n+1}^2 + 0.2t_{n+1}\right)$$
$$y_1 = 0.010, \ y_2 = 0.040, \ y_3 = 0.086$$
$$\textit{Jämför} \ \ y(0.1) = 0.010, \ y(0.2) = 0.040, \ y(0.3) = 0.090$$

Randvärdesproblem

- Ordinära differentialekvationer med randvärden som extra villkor kan på liknande sätt approximeras genom att ersätta derivator med dividerade differenser

$$y''(x) = f(x, y, y'), \quad a < x < b$$
$$y(a) = \alpha, \quad y(b) = \beta$$
$$\rightarrow \begin{cases} y_{n+1} - 2y_n + y_{n-1} = h^2 f(t_n, y_n, \frac{y_{n+1} - y_{n-1}}{2h}), & n = 1,2,..,N-1 \\ y_0 = \alpha, \quad y_N = \beta \end{cases}$$

- Det vi nu har är ett system av ickelinjära ekvationer som definierar de obekanta y_n. Systemet kan lösas med Newtons metod

Randvärdesproblem

- För linjära differentialekvationer fås ett system av linjära ekvationer som bestämmer lösningen, exempel

$$y''(x) = cy + f(x), \quad a < x < b$$
$$y(a) = \alpha, \quad y(b) = \beta$$
$$\rightarrow \begin{cases} y_{n+1} - 2y_n + y_{n-1} = h^2\left(y_n + f(x_n)\right), & n = 1,2,..,N-1 \\ y_0 = \alpha, \quad y_N = \beta \end{cases}$$
$$\begin{pmatrix} -2-ch^2 & 1 & 0 & . \\ 1 & -2-ch^2 & 1 & 0 \\ . & . & . & . \\ . & 0 & 1 & -2-ch^2 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ . \\ y_{N-!} \end{pmatrix} = \begin{pmatrix} h^2(f(x_2)-\alpha) \\ h^2 f(x_2) \\ \\ h^2(f(x_2)-\beta) \end{pmatrix}$$

Inskjutningsmetoden

- Ett randvärdes problem kan ersättas av ett begynnelsevärdesproblem med obekant begynnelsevärde $y'(a) = y(a;\xi) = \xi$ kopplat till en algebraisk ekvation

$$y''(x) = f(x,y,y'), \quad a < x < b$$
$$y(a) = \alpha, \quad y(b) = \beta$$
$$\rightarrow \begin{cases} y''(x) = f(x,y,y'), & x > a \\ y(a) = \beta, \quad y'(a) = \xi \\ y(b;\xi) = \beta \end{cases}$$

- En iterativ metod kan användas får att finna ξ som löser $y(b;\xi) = \beta$

System

- I likhet med begynnelsevärdessystem kan ode randvärdesproblem med högre derivator skrivas på första ordnings form

$$y''(x) = f(x, y, y'), \quad a < x < b$$
$$y(a) = \alpha, \quad y(b) = \beta$$
$$z = y' \rightarrow \frac{d}{dt}\begin{pmatrix} y \\ z \end{pmatrix} = \begin{pmatrix} z \\ f(t, y, z) \end{pmatrix}$$

- Randvillkoren skrivs ofta på formen

$$g(y(a), z(a), y(b), z(b)) = 0$$
$$\rightarrow \begin{pmatrix} y(a) - \alpha \\ y(b) - \beta \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

Matlab

- En lämplig standardmetod är ode45(fun,tspan,y_0)
  Den bygger på RK4 och och femte ordingens Runge-Kutta (RK5) i varje steg. Evaluering med två olika metoder användes för feluppskattning och för automatiskt val av steget h
- fun är högerledet i ekvationen – på skalär eller vektorform, tspan är vektor som börjar med t_0 och fortsätter med de t-värden där lösningen önskas, y_0 ar begynnelsevärdet - på skalär eller vektorform
- För styva problem kan ode15(fun,tspan,y_0) användas
- Notera att ett problem med högre derivator måste först skrivas som första ordnings system. Det gäller även för randvärdesproblem som kan approximaras av bvp4c(fun,bcfun,solinit), se matlabs hemsida för detaljer