KTH Engineering Sciences

Laboration 2: Ordinära differentialekvationer och randvärdesproblem

Målsättningen med denna laboration är att du förtärker din förståelse för olika metoder för numerisk lösning av differentialekvationer. Moment som behandlas är

- Modul 5: Begynnelsevärdesproblem för ODE-skalär och ODE-system
- Modul 6: Randvärdesproblem. Finita differensmetoden

Viktiga koncept som behandlas i laborationen är

- Stabilitet
- Noggrannhetsordning

Följande specifika kunskaper i PYTHON är bra att ha inför arbetet med laborationen

- Funktioner, vektorer, matriser och flerdimensionella fält.
- `for`-loopar och `while`-loopar
- Inbygga funktioner som `solve_ivp`

Tips: Tänk på att skriva Python-programmen/funktionerna på ett sådant sätt att det är enkelt att använda samma program/funktion för snarlika problem. Exempelvis bör ett program för Eulers metod enkelt kunna modifieras beroende på vilken differentialekvation man vill lösa.

F1) Vi vill lösa följande differentialekvation

$$\frac{dy}{dt} = f(t,y), \quad \text{där} \quad f(t,y) = 1 + t - y \quad \text{och} \quad y(0) = 1, \tag{1}$$

med Euler framåt fram till $t = T$, där $T = 1.2$.

a) Rita riktningsfältet för differentialekvationen (1) för $t \in [0\ T]$. Lägg sedan till den exakta lösningen i samma i samma diagram så att du kan jämföra den med fältet. Den exakta lösningen till ekvation (1) är $y(t) = e^{-t} + t$.

b) Skriv ett PYTHON-program som approximerar lösningen till ekvation (1) med Eulers metod-framåt och steglängden $h = 0.1$. Spara alla lösningsvärden (inklusive initialdata) i en vektor och plotta den numeriska lösningsvektorn som funktion av tiden. Skriv din kod så generellt som möjligt så att den går att återanvända för ett annat problem med annat högerled $f(t,y)$ och initialdata.

c) Verifiera att din lösning med Eulers metod-framåt vid tid $t = T$, ger felet $e_k = |y_k(T) - y_{exakt}(T)| \approx 0.0188$.

F2) Nu vill vi göra en konvergensstudie för problemet i **F1)**. Följ stegen nedan.

a) Börja med $h = 0.2$ och halvera h successivt fyra gånger så att $h = 0.2, 0.1, 0.05, 0.025, 0.0125$. För varje värde på h, beräkna en numerisk lösning med Euler framåt fram till sluttiden T. För varje h, spara lösningen vid sluttiden T, $y_k(T)$.

b) Beräkna felen i de numeriska lösningarna från **a)** enligt $e_k = |y_k(T) - y_{exakt}(T)|$. Verifiera att felen blir $[3.91 \cdot 10^{-2}, 1.88 \cdot 10^{-2}, 9.21 \cdot 10^{-3}, 4.56 \cdot 10^{-3}, 2.27 \cdot 10^{-3}]$ för $h = 0.2, 0.1, 0.05, 0.025, 0.0125$.

c) Felet e_k beter sig som $e_k \approx C_2 h^p$, där p är metodens noggrannhetsordning. När vi halverar h kommer därför kvoterna av två fel bli

$$\frac{e_{k,h}}{e_{k,h/2}} = 2^p.$$

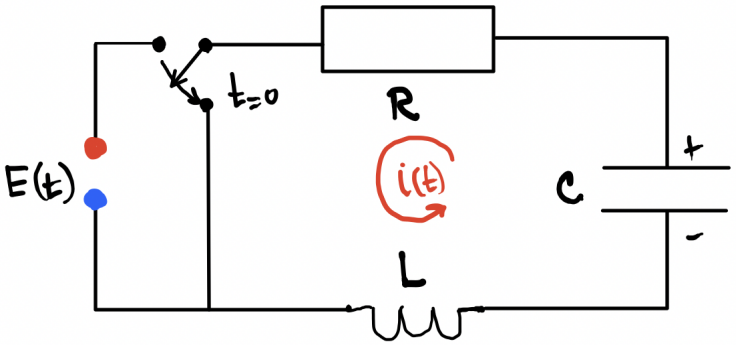
Vi kan då bestämma noggrannhetsordningen empiriskt genom

$$p \approx \log\left(\frac{e_{k,h}}{e_{k,h/2}}\right) \cdot \frac{1}{log(2)}.$$

Beräkna noggrannhetsordningen empiriskt genom formeln ovan och verifiera att de fyra beräknade nogrannhetsordningarna blir $p \approx [1.057, 1.028, 1.013, 1.007]$.

Tips! Försök att använda så lite koduppprepning som möjligt. Använd t ex en for-slinga där Euler framåt appliceras för ett h-värde i varje varv. Lagra numeriska lösningsvärden och fel i vektorer.

T1) Elektriskt system - högre ordningens differentialekvation. I denna uppgift ska vi lösa differentialekvationen för en RLC-krets, dvs en elektrisk krets med en kondensator och en spole som också har en viss resistans, R, se figur 1.



Figur 1: Dämpad svängningskrets

Vid tiden $t = 0$ sluter vi kretsen efter att ha laddat upp kondensatorn till ladningen Q_0. Kondensatorn kommer att urladdas, och vi ska bestämma approximativa lösningar för laddningens och strömmens tidberoende. Differentialekvationen för laddningens $q(t)$ är:

$$L\frac{d^2q}{dt^2} + R\frac{dq}{dt} + \frac{1}{C}q = 0, \quad q(0) = Q_0 \quad \frac{dq}{dt}(0) = 0 \tag{2}$$

där,

L står för spolens induktans, anges i Henry [H]

C står för kondensatorns kapacitans, anges i Farad [F]

R står för den totala resistansen i kretsen, anges i Ohms [Ω]

Strömmen $i(t)$ i kretsen då kondensatorn börjar laddas ur är:

$$i = \frac{dq}{dt}. \tag{3}$$

T1.a) Skriv om ekvation (2) till ett system av första ordningens differentialekvationer med hjälp av ekvation (3). Skriv systemet på formen

$$\mathbf{y}' = \mathbf{F}(t, \mathbf{y}) \quad \text{där} \quad \mathbf{y} = \begin{bmatrix} q \\ i \end{bmatrix}$$

T1.b) Skriv en PYTHON-funktion som tar tiden t, en vektor $\mathbf{y}$, och värden på R, L, C som inparametrar. Funktionen ska returnera vektorn $\mathbf{F}(t, \mathbf{y})$. Funktionen ska vara skriven så att den går att använda med PYTHONs inbyggda ode-lösare solve_ivp

T1.c) Lös systemet av differentialekvationer från **a)** med PYTHONs inbyggda funktion solve_ivp och metoden RK45 på tidsintervallet $t = [0, 20]$ med

i) $Q_0 = 1, L = 2, C = 0.5, R = 1$ för dämpad svängning

ii) $Q_0 = 1, L = 2, C = 0.5, R = 0$ för odämpad svängning

T1.d) Dela nu in tidsintervallet $t = [0, 20]$ i $N = 20, 40, 80, 160$ ekvidistanta tidssteg. För varje värde på N, lös systemet av differentialekvationer från **a)** med Euler framåt. Använd $L = 2, C = 0.5, R = 1$ (dämpad svängning). Plotta lösningen för varje värde på N.

För vilka värden på N blir den numeriska lösningen stabil (dvs lösningen växer ej)? För vilka värden på N blir den numeriska lösningen instabil? Notera att en numerisk lösning kan vara stabil men ändå inte noggrann (använd lösningen från `solve_ivp` som referenslösning, dvs för att approximera den exakta lösningen).

T1.e) För fallet med dämpad svängning, utför en konvergensstudie för Euler framåt och bestäm noggrannhetsordningen för metoden empiriskt. Beräkna felet komponentvis, se förklaring nedan, vid sluttiden $T = 20$ och använd lösningen från `solve_ivp` som referenslösning. **Gör så här**: Börja med ett värde på N som leder till en stabil numerisk lösning. Dubblera sedan N (halvera tidssteget h) successivt och beräkna felen (ett fel per komponent i lösningen) för varje värde på N. Följ stegen i **F2 c)** för att beräkna noggrannhetsordningen komponentvis.

T2) Termiskt system. Temperatur i stav - randvärdesproblem. Temperaturfördelningen i en cylindrisk stav med längden L beskrivs av differentialekvationen

$$\begin{aligned} k\frac{d^2T}{dx^2} &= q(x), \quad 0 < x < L, \\ T(0) &= T_L, \\ T(L) &= T_R, \end{aligned} \tag{4}$$

där k är stavens värmeledningsförmåga, $q(x)$ är en källterm med värme som tillförs. Randvillkoren beskriver att stavens ändpunkter hålls vid fixa temperaturer T_L och T_R.

Differentialekvationen med randvillkor (4) kan lösas numeriskt med finita differensmetoden genom att diskretisera intervallet $x = [0, L]$ i N delintervall enligt $x_j = hj$, $j = 0, 1, 2, \cdots, N$, där längden av varje delintervall ges av steglängden $h = \frac{L}{N}$. Vi får då en approximativ lösning $T_j \approx T(x_j)$ i punkterna x_j, $j = 1, 2, \cdots, N-1$. Lösningen i randpunkterna x_0 och x_N uppfylls exakt av randvillkoren, dvs $T_0 = T(0) = T_L$, $T_N = T(L) = T_R$. Låt $L = 1$, $k = 2$, $q(x) = 50x^3 \ln(x + 1)$, $T_L = T_R = 2$.

T2.a) Diskretisera (4) med centrala finita differenser för $N = 4$. Skriv ut systemmatrisen och högerledet med alla element.

T2.b) Diskretisera (4) med centrala finita differenser för ett generellt N. Strukturen för systemmatrisen och högerledet ska framgå tydligt.
Tips: Använd generella benämningar på L,k, q(x) osv både i diskretiseringen och i koden så att du enkelt kan modifiera koden för att lösa ett liknande problem.

T2.c) Skriv en PYTHON-funktion `diskretisering_temperatur` som returnerar systemmatrisen A (som en gles matris, dvs inte full) och högerledet HL (inklusive randvillkor) givet ett funktionshandtag för $q(x)$ och värden på k, randvillkoren T_L, T_R och antal diskretiseringsintervall N. Ett anrop till funktionen kan se ut så här:
`A, HL = diskretisering_temperatur(N, q, k, Tl, Tr)`
Du kan testa din funktion genom att sätta $N = 4$ och verifiera att

$$A = 32 \cdot \begin{bmatrix} -2 & 1 & 0 \\ 1 & -2 & 1 \\ 0 & 1 & -2 \end{bmatrix}, \quad HL \approx \begin{bmatrix} -63.826 \\ 2.534 \\ -52.196 \end{bmatrix}.$$

T2.d) Lös randvärdesproblemet med $N = 100$ och plotta temperaturen som funktion av x. Vad blir den approximativa temperaturen vid $x = 0.2$?

T2.e) Gör en konvergensstudie med hjälv av steglängdshalvering och studera hur den approximativa temperaturen i punkten $x = 0.7$ ändras när N ökar. Verifiera att metodens noggrannhetsordning stämmer med teorin. Du kan t ex starta med $N = 50$ och dubblera N successivt. Temperaturen i punkten $x = 0.7$ bör konvergera mot 1.6379544 (avrundat värde).
Tips: Om noggrannhetsordningen blir lägre än förväntat kan det vara så att du mäter temperaturen i fel punkt, t ex $x = 0.7 + h$ eller $x = 0.7 - h$. Var noggrann med att välja ut rätt element ut lösningsvektorn.

T2.f) Testa att ändra randvillkoren så att ändarna har olika temperaturer, dvs $T_L \neq T_R$. Notera hur stor påverkan randvillkoren har på temperaturfördelningen i staven.

Inför redosvisning

Samtliga program bör kontrolleras så att de exekveras felfritt innan redovisningen. Dessa kontroller kan med fördel genomföras i ett tidigt skede, dels som förberedelse inför laborationen, dels för att säkerställa funktionaliteten i samtliga uppgifter. Inför den muntliga redovisningen gäller att samtliga Python-filer som har använts ska vara tillgängliga på den dator som används vid redovisningen. Filerna ska vara tydligt namngivna för att möjliggöra en effektiv åtkomst och körning. Samtliga filer ska dessutom laddas upp via Canvas före redovisningstillfället. Under redovisningen ska ni, individuellt, kunna redogöra för den teori och de algortimer ni har använt. Ni ska också kunna svara på frågor som ställs under redovisningen och förklara hur era Python-program fungerar. **Kom väl förberedda!**