KTH Engineering Sciences

Numerisk lösning av integraler och differentialekvationer.

Målsättningen med denna laboration är att du ska få förståelse för olika metoder för numerisk lösning av integraler och differentialekvationer. Metoder som behandlas är

- Simpsons regel
- Eulers metod
- Runge-Kutta
- Finita differensmetoden

Viktiga koncept som behandlas i laborationen är

- Approximationsfel
- Noggrannhetsordning

Innan ni börjar med uppgifterna:

- **Anmäl er (bägge i gruppen)** till en av labbgrupperna SF1546 - **Lab 2** 1–150.
- Gör de obligatoriska förberedelsuppgifterna i Matlab Grader.
- Ni ska skicka in de MATLAB-program och egenskrivna funktioner som efterfrågas nedan. Egenskrivna funktioner lägger ni längst ner i huvudprogrammet.
- Deadline för **inlämning av filer är tisdag 2 mars**. Se till att ni har skickat in allt som efterfrågas.
- Redovisningen äger rum den **4 mars 2021**.
- Vid redovisningen ska ni kunna svara på frågor om teori och algoritmer som ni använt.

1. Numerisk integration

Den totala massan av ett ämne som transporteras genom ett rör över en tidsperiod, t_0 till t_1 kan beräknas som

$$M = \int_{t_0}^{t_1} Q(t)c(t)dt \tag{1}$$

där M är massan [mg], $Q(t)$ är flödeshastigheten [m^3/min] och $c(t)$ är koncentrationen [mg/m^3] av det aktuella ämnet. Tiden mäts i minuter.

Följande två funktioner specificerar hur flödet respektive koncentrationen beror av tiden

$$Q(t) = 9 + 5\cos^2(0.4t)$$
$$c(t) = 5e^{-0.5t} + 2e^{0.15t}$$

Bestäm den totala massan som transporterats i röret mellan $t_0 = 3$ min och $t_1 = 9$ min genom att beräkna integralen numeriskt med Simpsons formel.

UPPGIFT 1. SIMPSONS FORMEL

a) Skriv ett MATLAB-program `Simpson.m` som beräknar värdet av integralen (1) med Simpsons formel för $n = 8$ och $n = 16$ där n är antalet delintervall. Programmet ska skriva ut värdet på integralen för bägge värdena på n.

b) Hur många decimaler verkar tillförlitliga (använd `format long`)?

Definiera approximationsfelet som

$$E_h = |M - M_h| \tag{2}$$

där M_h är integralapproximationen beräknad med steglängd h och M är en mycket noggrann referenslösning med Simpsons metod.

c) Undersök hur E_h beror på steglängden h genom att utvidga programmet så att det beräknar integralvärdet M_h, för en följd av värden på steglängden h. Beräkna approximationsfelet enligt (2) och plotta E_h som funktion av h. Använd MATLABs plotkommando `loglog`. Uppskatta metodens noggrannhetsordning med hjälp av figuren [1].

Ett annat sätt att uppskatta en metods noggrannhetsordning är att beräkna

$$\frac{M_h - M_{h/2}}{M_{h/2} - M_{h/4}}. \tag{3}$$

för ett antal värden på h (fler än 3).

d) Uppskatta metodens noggrannhetsordning genom att använda värdena på M_h från c) för att beräkna och skriva ut kvoter enligt (3) [1].

[1] Se föreläsningsanteckningarna om noggrannhetsordning.

Skicka in programmet `Simpson.m`

2. Numerisk lösning av ODE - Pendeln

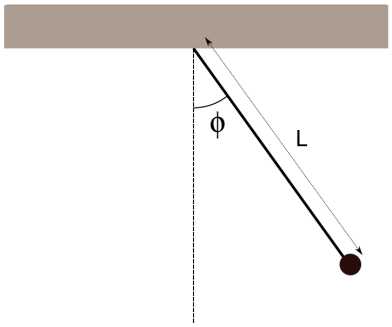
Vi studerar en dämpad pendeln som beskrivs av den ordinära differentialekvationen (ODEn)

$$m\phi'' + \mu\phi' + \frac{mg}{L}\sin(\phi) = 0, \qquad (4)$$

med begynnelsedata

$$\phi(0) = \phi_0, \quad \phi'(0) = 0.$$

Den sökta funktionen $\phi(t)$ är vinkeln för pendelns utslag. Parametrarna i ekvationen är pendelns massa m [kg], dämpningskonstanten μ [N/m], tyngdaccelerationen g [m/s^2] och pendelns längd L [m]. Se figur. I uppgifterna nedan ska ni använda värdena $m = 0.6$, $L = 1.5$, $g = 9.81$ och $\mu = 0.2$.



UPPGIFT 2. FRAMÅT EULER

a) Skriv om (4) till ett system av första ordningens differentialekvationer.

b) Skriv ett MATLAB-program `Eulerfram.m` som löser ODE-systemet i a) med Framåt Euler till tiden $t = 5$. Använd tidssteget $h = 0.01$ och $\phi_0 = 0.5$ Plotta lösningen $\phi(t)$ på intervallet $0 \leq t \leq 5$.

   För att få ett effektivt och överskådligt program ska ni skriva en funktion `feuler` för Eulers metod för det aktuella problemet som programmet `Eulerfram.m` ska använda sig av. Funktionen ska ta följande indataargument:

   - begynnelsedata `u0` (en kolumnvektor med två komponenter),
   - sluttiden `T`,
   - antal tidssteg `n` (notera: $h = T/n$).

   Funktionen ska returnera två variabler:

   - kolumnvektorn `t` som innehåller alla tidpunkter, dvs `t = [0,h,2h, ..., T]`^{T} (längd `n+1`),
   - matrisen `y` som innehåller två kolumner med approximationerna av ϕ respektive ϕ' i motsvarande tidpunkter (storlek $n + 1$ rader, 2 kolumner).

   Notera: Med givna returvärden `t` och `y` från `feuler.m` kan lösningsplotten nu erhållas med kommandot `plot(t,y(:,1))`.

   Tips: För det exakta värdet på $\phi(t)$ gäller att $0.15 < \phi(5) < 0.25$. Den numeriska lösningen med $h = 0.01$ hamnar precis utanför intervallet. (Med tillräckligt litet h kommer den dock ligga i intervallet.)

c) Verifiera att er implementation av Framåt Euler på problemet har noggrannhetsordning ett genom att beräkna $\phi(5)$ för flera olika `n`. Beräkna kvoter av differenser mellan lösningar med n, $2n$, $4n$, $8n$... på samma sätt som i uppgift 1. Programmet ska skriva ut en tabell med kvoter och motsvarande noggrannhetsordningar. Här kan man exempelvis använda funktionen `table`. Skriv `help table` i MATLAB för mer information.

Skicka in programmet `Eulerfram.m` och funktionsfilen `feuler.m`

UPPGIFT 2. RUNGE-KUTTA 4 OCH ODE45

d) Skriv ett MATLAB-program `RungeKutta4.m` som löser ODE-systemet i a) men med Runge-Kutta 4. Beräkna lösningen med tidssteget $h = 0.01$ till tiden $t = 5$. Låt $\phi_0 = 0.5$ Plotta lösningen $\phi(t)$ på intervallet $0 \leq t \leq 5$.

Även i denna uppgift ska ni skapa en funktion `RK4` för Runge-Kutta 4 för det aktuella problemet och använda er av funktionen i programmet `RungeKutta4.m`. Funktionen ska ha samma inargument och returvärden som för Eulers metod.

e) Verifiera att er implementation av Runge-Kutta 4 är korrekt genom att kontrollera noggrannhetsordningen på samma sätt som i c). Programmet ska skriva ut en tabell med kvoter och motsvarande noggrannhetsordningar. Stämmer noggrannhetsordningen med det teoretiska värdet för Runge-Kutta 4?

f) Slutligen ska ni använda MATLABs inbyggda funktion `ode45`. Skriv ett (kort) program `Testaode45.m` där ni använder `ode45` för att lösa ODE-systemet i a). Programmet ska beräkna lösningen till tiden $t = 5$. Låt $\phi_0 = 0.5$. Programmet ska även plotta lösningarna $\phi(t)$ och $\phi'(t)$ på intervallet $0 \leq t \leq 5$.

Skicka in programmen `RungeKutta4.m` och `Testaode45.m` samt funktionsfilen `RK4.m`

3. Randvärdesproblem

Följande differentialekvationsproblem beskriver temperaturfördelningen $T(x)$ i en stav av längden L [m].

$$-\frac{d}{dx}\left(k\frac{dT}{dx}\right) = Q(x), \qquad T(0) = t_0, \quad T(L) = t_1. \tag{5}$$

Vänster och höger ändpunkt på staven har den konstanta temperaturen t_0 resp t_1 [K]. Konstanten k [N/(K· s)] är stavens värmeledningsförmåga och $Q(x)$ [N/(m^2·s)] är den värmemängd som per tidsenhet och volymsenhet genereras i staven, tex genom radioaktivitet eller yttre uppvärmning. Antag att $L = 3$, $k = 2$, $t_0 = 290$, $t_1 = 400$ och $Q(x)$ är funktionen

$$Q(x) = q_0\, e^{-q_1(x-0.7L)^2} + 200, \qquad q_0 = 3000, \quad q_1 = 200.$$

Differentialekvation och randvillkor (5) kan lösas numeriskt med hjälp av finita differensmetoden. Diskretisera intervallet $[0, L]$ enligt $x_i = ih$, $i = 0, 1, 2, \ldots, n, n+1$, där $h(n+1) = L$ och låt $T_i \approx T(x_i)$. Efter en approximation av andraderivatan med centraldifferens erhålles

$$\frac{-T_{i-1} + 2T_i - T_{i+1}}{h^2} = \frac{1}{k}Q(x_i), \qquad i = 1, 2, \ldots, n. \tag{6}$$

Tillsammans med randvillkoren $T_0 = t_0$ och $T_{n+1} = t_1$ leder diskretiseringen till ett linjärt ekvationssystem

$$AT = b,$$

där A är en $n \times n$-matris, T är en n-vektor med temperaturvärdena i det inre av intervallet och b är en n-vektor som beror på randvärdena t_0, t_1 och $Q(x_i)$-värdena.

UPPGIFT 3. FINITA DIFFERENSMETODEN

a) Skriv ett MATLAB-program `Temperatur.m` som konstruerar matrisen A och högerledet b för en given storlek n. Matrisen kommer endast att ha ett fåtal nollskilda element. Matrisen skapar du tex med hjälp av MATLABs kommando `diag(v)`, som genererar en diagonalmatris med vektorn `v` på diagonalen, samt generaliseringen `diag(v,p)` som genererar en matris med vektor `v` på diagonal nummer `p`, där `p` kan vara både positiv och negativ.

Tips: Skriv först ner matrisen A med papper och penna för ett enkelt fall, tex $n = 4$, för att se strukturen hos matrisen. Verifiera också att programmet genererar korrekt matris för detta fall.

b) Utöka programmet så att det räknar ut temperaturen T genom att lösa det linjära ekvationssystemet $AT = b$ med backslash, \. Pröva först med $n = 10$ inre punkter. Lösningsvektorn T kommer att innehålla temperaturen i alla punkter x_i *förutom* i randpunkterna $x_0 = 0$ och $x_{n+1} = L$. Lägg till dessa randvärden genom att bygga ut T-vektorn med två element. Skapa motsvarande x-vektor och plotta temperaturen som funktion av x på *hela* intervallet $0 \leq x \leq L$.

c) Utöka programmet ytterligare så att det beräknar temperaturen för $n = 40, 80, 160, 320$. Programmet ska dels plotta de erhållna resultaten i samma figur (fyra kurvor), dels skriva ut den minimala och maximala temperaturen, samt medeltemperaturen för varje n. Formatera utskriften som en tabell med fyra rader ($n = 40, 80, 160, 320$) och fyra kolumner (n, min T, max T och medel T).

Tips: När $N = 40$ så blir max $T = 577.7308$ (avrundat).

- Hur tillförlitliga är de beräknade värdena? Uppskatta felet i den numeriska lösningen med $N = 320$. (Läs om tillförlitlighetsbedömning i det sista avsnittet i anteckningarna om noggrannhetsordning.)

d) Nu ska du göra en störningsanalys där du beräknar osäkerheten i den maximala temperaturen när osäkerheten i q_0 och q_1 är given av $q_0 = 3000 \pm 200$ och $q_1 = 200 \pm 10$. Använd experimentell störningsräkning.

För att lösa uppgiften ska du skapa en funktion `stav.m` som tar värden på q_0 och q_1 som indata, beräknar lösningen som i b) och returnerar maximala temperaturen.

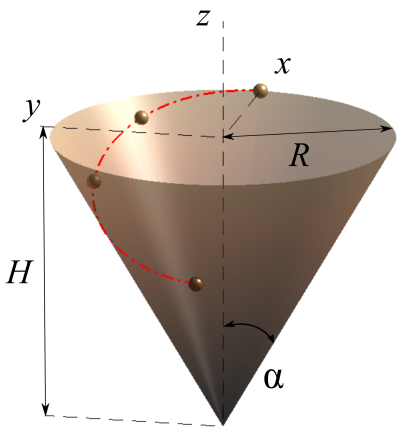
Skriv sedan ett program `Temperatur2.m` som gör en experimentells störningsanalys. Programmet ska skriva ut maximala temperaturen och dess osäkerhet.

Välj ett n för vilket de numeriska felen är betydligt mindre än osäkerheterna. (Läs om experimentell störningsräkning i anteckningarna om felanalys.)

Skicka in programmen `Temperatur.m` och `Temperatur2.m` samt funktionen `stav.m`.

Kula i kon - Frivillig uppgift

Betrakta en kula som rör sig inuti en öppen kon med höjden $H = 1.8$ [m], spetsvinkeln $\alpha = \pi/5$ och toppradien $R = H\tan\alpha$ [m]. Kulan ges en horisontell hastighet v_0 [m/s] vid konens övre kant och rullar sedan runt på konens insida. Låt (r, θ, z) vara kulans position i cylinderkoordinater. Dess rörelse beskrivs då av två kopplade andra ordningens ODE för r och θ,



$$r'' = f_1(r, \theta, r', \theta'), \qquad \theta'' = f_2(r, \theta, r', \theta'), \tag{7}$$

där

$$f_1(r, \theta, r', \theta') = -\frac{N}{m}\left(\cos\alpha + \frac{\mu r'}{v}\right) + r\,(\theta')^2\,,$$
$$f_2(r, \theta, r', \theta') = -\frac{\mu N \theta'}{mv} - \frac{2r'\theta'}{r}.$$

Här är v [m/s] kulans hastighet och N [N] den normalkraft som verkar på kulan, givna av

$$v(r, r', \theta') = \sqrt{(r')^2 + (r\theta')^2 + (r'/\tan\alpha)^2}, \qquad N(r, \theta') = mr(\theta')^2\cos\alpha + mg\sin\alpha. \tag{8}$$

Parametrarna i differentialekvationerna är kulans massa m [kg], tyngdaccelerationen g [m/s^2] och den dimensionslösa friktionskoefficienten μ. Begynnelsedata ges av $r(0) = R$, $\theta(0) = 0$, $r'(0) = 0$ och $\theta'(0) = v_0/R$.

ODEerna (7) kan skrivas om som ett system av fyra första ordningens ODE. Genom att sätta $u_1 = r$, $u_2 = \theta$, $u_3 = r'$ och $u_4 = \theta'$ får vi

$$\frac{d\mathbf{u}}{dt} = \mathbf{F}(\mathbf{u}), \qquad \mathbf{u} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}, \qquad \mathbf{F}(\mathbf{u}) = \begin{bmatrix} u_3 \\ u_4 \\ f_1(u_1, u_2, u_3, u_4) \\ f_2(u_1, u_2, u_3, u_4) \end{bmatrix}. \tag{9}$$

Funktionen $\mathbf{F}(\mathbf{u})$ finns implementerad i filen `fkula.m` som ligger i Canvas. Parametervalen där är $m = 1$, $g = 9.81$ och $\mu = 0$ (ingen friktion). Observera att funktionen tar två argument, trots att t-parametern inte används. Detta beror på att MATLABs inbyggda ODE-funktioner kräver ett sådant format.

BERÄKNA KULANS BANA MED RUNGE-KUTTA 4 OCH `ode45`

a) Skriv ett MATLAB-program `kula.m` som beräknar kulans bana genom att lösa (9) med Runge–Kutta 4. Programmet ska anropa funktionen `fkula.m`.

Bestäm lösningen med $v_0 = 0.9$ fram till tiden $t = 5$. Använd steglängden $h = 0.05$. Programmet ska sedan plotta kulans x-, y- och z-koordinat som funktion av tiden t i samma figur (tre kurvor). Notera att den framräknade lösningen ges i cylinderkoordinater. Kulans position i kartesiska koordinater ges via sambanden

$$x = r\cos\theta, \qquad y = r\sin\theta, \qquad z = r/\tan\alpha. \tag{10}$$

För att få en bild av hur kulan rör sig i konen kan ni med hjälp av funktionen `plotkula.m` som finns i Canvas rita upp kulans bana i 3D. Funktionen anropas som `plotkula(x,y,z)`

där `x`, `y` och `z` är kulans x-, y- och z-koordinater som ni beräknat ovan. Prova även att animera kulan med anropet `plotkula(x,y,z,p)` där argumentet `p` styr hur snabb animeringen är. Notera att kulan inte riktigt beter sig som man skulle tro. Detta beror på att friktionen är satt till 0.

b) Bygg ut programmet så att det också beräknar kulans bana med MATLABs inbyggda funktion `ode45` för samma fall som i deluppgift a). Programmet ska sedan plotta banans projektion på xy-planet (dvs x-vektorn plottas mot y-vektorn) för denna lösning och för er Runge-Kutta-lösning, i samma figur (två kurvor).

   – Blir det någon synlig skillnad mellan de två lösningarna?

c) Lägg nu på lite friktion genom att ändra $\mu = 0$ till $\mu = 0.05$ i `fkula.m`. Gå igenom experimenten i deluppgifterna a)–b) igen. Hur ändras kulans bana?

Rörelsemängdsmoment och energi

Kulans rörelsemängdsmoment L [N·m·s] och dess totala energi E [N·m] ges av uttrycken

$$L = mr^2\theta', \qquad E = \frac{mv^2}{2} + mgz. \tag{11}$$

I mekaniken visas att båda dessa storheter i det aktuella fallet bevaras i tiden, dvs de är konstanta.

Det är viktigt att även den numeriska lösningen bevarar den här typen av *invarianter* väl för att kunna ge fysikaliskt korrekta förutsägelser. Det kan vara svårt när simuleringstiden är lång och ofta behöver man använda speciella numeriska metoder i det fallet.

Ni ska nu undersöka hur bra metoderna ovan är på att bevara L och E.

BERÄKNA KULANS RÖRELSEMÄNGDSMOMENT OCH ENERGI

d) Skriv ett MATLAB-program `kula2.m` som löser (9) fram till $t = 125$ på följande tre sätt

   1. Runge–Kutta 4 med $h = 0.05$,
   2. Runge–Kutta 4 med $h = 0.01$,
   3. `ode45`.

   För varje fall ovan ska L och E beräknas. Notera att från en numerisk lösning kan ni beräkna L och E med hjälp av (11) och uttrycken för v och z i (8) respektive (10).

   Programmet ska sedan plotta L som funktion av $t \in [0, 125]$ för alla tre fallen i samma figur (tre kurvor) och E som funktion av t för alla tre fallen i en annan figur (tre kurvor).

   Använd `legend`-kommandot för att indikera vilken kurva som är vilken.

   – Vilken metod gör bäst ifrån sig här? Tag hänsyn både till noggrannhet (hur konstant är E och L) och beräkningskostnad (antal tidssteg[1] metoderna använt).

   – Hur bra tror ni Framåt Euler skulle vara för det här problemet?