SF1663/4 HT2012 för T
NA-KTH
1 november 2012

Laboration 4

Datastrukturer, ordinära differentialekvationer och glesa system

Sista dag för bonuspoäng, se kursplanen. Kom väl förberedd och med välordnade papper till redovisningen. Numeriska resultat ska finnas noterade. Båda i laborationsgruppen ska kunna redogöra för teori och algoritmer!

Efter den här laborationen skall du känna igen problemtyperna randvärdes- och begynnelsevärdesproblem för ordinära differentialekvationer och kunna lösa dessa med differensmetoder. Du skall kunna analysera noggrannhetsordning och bestämma stabilitetsegenskaper både teoretiskt och experimentellt. Du skall också lära dig att hantera vektorer och lösa stora linjära ekvationssystem.

1. Kryptering

Läsanvisning: PEng Kap X

Krypering har blivit en del av vår vardag även om vi inte tänker på det. När du loggar in på din personliga banksida eller pratar i mobiltelefon så sker kommunikationen krypterat. I den här laborationen ska du skriva en applikation som analyserar en krypterad text och sedan dekrypterar den.
En text som är på engelska finns lagrad på en fil som ska läsas in av ditt program. Det finns många olika algoritmer för kryptering men texten förutsätts ha krypterats med s k Caesar-rullning vilket innebär att varje bokstav har förskjutits ett visst antal bokstäver framåt. Med t.ex. två bokstävers förskjutning blir "*The zebra has stripes*" krypterat till "*Vjg bgdtc jcu uvtkrgu*".
Men, med vetskap om att bokstaven 'e' är den vanligast förekommande i engelskan kan texten dekrypteras med hjälp av ett frekvensdiagram! Om t.ex. bokstaven 'g' är den mest frekventa bokstaven i den krypterade texten betyder det att den troligtvis har förskjutits två steg framåt i alfabetet.
Ditt program behöver inte koda några andra tecken än engelska bokstäver, därför kan (och bör!) dessa andra tecken direkt släppas igenom vid inläsningen. Frekvensanalysen redovisas i form av ett tårtdiagram som endast ska innehålla de tio mest frekventa bokstäverna. Det blir dessutom enklare om du vid inläsningen gör om alla bokstäver i texten till versala. På kurshemsidan finns flera krypterade filer att utgå ifrån när du testar ditt program.

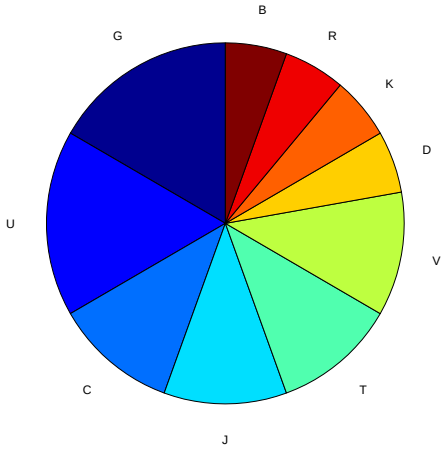
1a. Avkryptering

Skriv ett MATLAB-program som frågar efter en krypterad fil, frekvensanalyserar den, visar ett tårtdiagram med de 10 vanligaste bokstäverna, efterfrågar rullbokstav (rullbokstaven skall kunna väljas fritt) och sedan avkrypterar och skriver ut texten i MATLAB-fönstret. Radindelning och skiljetecken skall bibehållas.

Dialogen och tårtdiagrammet kan tex se ut som:

```
Vad heter den krypterade filen?  kryp99.txt
Vilken bokstav har man rullat e till? g
Avkrypterat:

The zebra has stripes
```



1b. Frivillig uppgift: Versaler och gemener

I förra uppgiften var det tillåtet att omvandla alla bokstäver till stora bokstäver. En bättre version av programmet skulle klara att bevara indelningen med små och stora bokstäver. Kan du göra det?

2. Begynnelsevärdesproblem

Läsanvisning: GNM Kap 6.1-6.2

a) Givet är följande differentialekvation

$$\frac{dy}{dt} = \cos(t) - 5y, \quad y(0) = 0.5, \quad t \in [0, 15].$$

I denna deluppgift ska ovanstående differentialekvation lösas numeriskt med Eulers framåtmetod, Eulers bakåtmetod och trapetsmetoden. Uppgiften går ut på att undersöka

- hur trunkeringsfelet avtar med steglängden h (för små värden på h), dvs noggrannheten
- hur den numeriska lösningen uppför sig för 'stora' h-värden (stabilitet)

(a) Lös först differentialekvationen exakt (analytiskt), dvs med metoder som du lärt dig i matematik. Denna lösning betecknas $y(t)$.

(b) Dela in tidsintervallet $[0, 15]$ i n ekvidistanta steg h. Eulerlösningen i en punkt t skattad med steglängden h betecknas $y_E(t, h)$. Skriv ett Matlabprogram som beräknar Eulerlösningarna för $n = 150$, $n = 300$, $n = 600$ och $n = 1200$. Plotta i samma graf den exakta lösningen $y(t)$ samt de de tre Eulerlösningarna. Plotta även i ett loglog-diagram felet i slutpunkten, dvs $|y(15) - y_{Euler}(15, h)|$ som funktion av h. Vilken ordning hos Eulers framåtmetod kan utläsas ur diagrammet? Stämmer det med teorin?

(c) Gör samma beräkningar, grafer, diagram och dra slutsatser för Eulers bakåt-metod.

(d) Gör samma beräkningar, grafer, diagram och dra slutsatser för trapetsmetoden.

(e) Dela in tidsintervallet $[0, 15]$ i $n = 10, 20, 40$ och 80 ekvidistanta steg. Beräkna de fyra Eulerlösningarna och plotta dem tillsammans med den exakta lösningen i varsin graf.
Gör samma sak med Eulers bakåt-metod.
Gör samma sak med trapetsmetoden.
Vilken slutsats kan du dra beträffande den numeriska stabiliteten för de tre metoderna genom att titta på graferna? För den eller de icke-stabila, undersök vilken steglängd h går gränsen (ungefär)? (Tips: prova n mellan 30 och 50 med Euler framåt).

3. Högre ordningens differentialekvation

Läsanvisning: GNM Kap 6.1C och 4.1E
Följande andra ordningens differentialekvation beskriver en pendels rörelse.

$$\frac{d^2\phi}{dt^2} + \frac{g}{L}\sin(\phi) = 0, \quad \phi(0) = \frac{\pi}{2.5}, \quad \frac{d\phi}{dt}(0) = 0, \quad t = [0, T]$$

Här är ϕ vinkeln mot lodlinjen och ϕ' vinkelhastigheten. Längden på snöret är $L = 4\ m$ och tyngdkraften $g = 9.81\ m/s^2$.

(a) Skriv om differentialekvationen som ett system av första ordningens differentialekvationer. Systemet skall redovisas på papper.

(b) Lös systemet med MATLABs inbygga ode-lösare `ode45`. Välj ett tidsintervall, $[0, T]$, som gör att pendeln hinner svänga lite drygt två hela perioder. Plotta vinkel och vinkelhastighet som funktion av tiden.

(c) Animera pendelns gång. Animeringen kan göras med tex ett anrop av följande MATLAB-kod:

```
function anim(tut,fiut,L);
   for i=1:length(tut)-1
      x0=L*sin(fiut(i));y0=-L*cos(fiut(i));
      plot([0,x0],[0,y0],'-o')
      axis('equal')
      axis([-1 1 -1 0]*1.2*L)
      drawnow
      pause(tut(i+1)-tut(i))
   end;
end
```

där `tut` är tidpunkterna vid vilka `ode45` har räknat ut lösningen, `fiut` är den uträknade vinkeln vid motsvarande tidpunkt och `L` är pendelns längd.

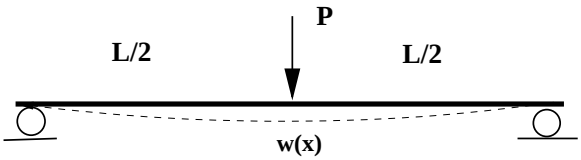
(d) Bestäm pendelns svängningstid (dvs period) med minst 1 decimal genom att göra interpolation i den beräknade pendelrörelsen och bestämma interpolationspolynomets nollställe. (Tips: Välj ett lämpligt interpolationspolynom! Ledning: över vilket intervall bör interpolationspolynomet gå?)

(e) Beror svängningstiden av L?

4. Randvärdesproblem och stora linjära ekvationssystem

Läsanvisning: GNM Kap 6.3 och 4.1 (glesa matriser).

En homogen balk av höghållfast stål är fritt upplagd horisontellt på två rullstöd med avståndet



$L = 2.45\ m$. Balken har ett cirkulärt tvärsnitt med en radie $r = 2.90 \cdot 10^{-2}\ m$ och en elasticitetsmodul $E = 2.70 \cdot 10^{11}\ N/m^2$. Mitt på balken verkar en nedåtriktad kraft $P = 260\ N$. Man vill beräkna balkens utböjning, $w(x)$.
För ett tvärsnitt vid läget x finns ett samband mellan momentet $M(x)$ och balkens utböjning $w(x)$. Om kraften P är tillräckligt liten ges detta samband av följande linjära modell

$$\frac{M(x)}{EI} = -w''(x). \tag{1}$$

I är yttröghetsmomentet för balken och ges utav formeln

$$I = \frac{\pi r^4}{4}$$

Momentet beror av kraften P enligt

$$M(x) = \begin{cases} -\frac{Px}{2} & x \leq \frac{L}{2} \\ -\frac{PL}{2}(1 - \frac{x}{L}) & x \geq \frac{L}{2} \end{cases}$$

Randvillkoren, att balken är fritt upplagd, kan skrivas som

$$w(0) = 0, \quad \text{och} \quad w(L) = 0. \tag{2}$$

Ekvation (1) tillsammans med randvillkoren (2) kan lösas numeriskt med hjälp av finita differensmetoden. Diskretisera balken i N delar (dvs $N+1$ punkter), $x_j = jh,\ j = 0, 1, \ldots N$, $h = L/N$, och w_j är utböjningen i punkten $x = x_j$. Använd en andra ordningens noggrann differens-approximation. Diskretiseringen leder till ett linjärt ekvationssystem

$$A\mathbf{w} = \mathbf{f} \tag{3}$$

där A är en matris och $\mathbf{w} = (w_1, w_2, \ldots)$ är en vektor med de obekanta utböjningarna.

a) Härled och skriv ner matrisen A och vektorn f med papper och penna. Vilken struktur har matrisen A? Vilken dimension har matrisen A? Vilken dimension har vektorn f?

b) Skriv ett Matlab-program som löser randvärdesproblemet. Matrisen A kommer endast att ha ett fåtal nollskilda element. (Tips: Skapa matrisen i Matlab med Matlabs `diag` och `eye`).

Räkna ut utböjningen w för fallet $N = 200$. Lösningen får du genom att lösa det linjära ekvationssystemet $A\mathbf{w} = \mathbf{f}$. Du behöver också räkna ut $\mathbf{f}$ som ges av högerledet i ekvation (3) och sambanden för momentet, $M(x)$, och yttröghetsmomentet, I. Använd värdena på L, E, P och r enligt ovan. Lösningen, $\mathbf{w}$, kommer att innehålla utböjningen i alla punkter, x_j (eventuellt utom i randpunkterna där $w = 0$).
Plotta utböjningen som funktion av x för $0 \leq x \leq L$. Vilket värde på maximala utböjningen fås med $N = 250$? Ge en skattning på trunkeringsfelet i ditt värde för maximala utböjningen du fick med $N = 200$.

c) Bestäm balkens maximala utböjning med (minst) 6 korrekta decimaler.

d) Föreslå ett lämpligt minsta antal delar, N, att dela upp stången i om man söker utböjningen i $x = 1.33$?

e) Backslash-kommandot i MATLAB använder vanlig Gauss-eliminering för att lösa ekvationssystemet. Undersök hur tidsåtgången för gausseliminering beror av systemmatrisens storlek genom att lösa $A\mathbf{w} = \mathbf{f}$ för olika N, tex $N = 100, 200, 400, 800$. För att mäta tiden kan MATLAB-funktionen `cputime` eller `tic` och `toc` användas (`help` ger mer information). För att få en bra noggrannhet av mätningen av cpu-tiden (eftersom den är kort) bör man kanske upprepa beräkningarna (dvs lösandet av det linjära ekvationssystemet) i en for-slinga (tex 5, 20 eller 100 gånger, beroende på dator) och sedan ta medelvärdet. Plotta tidsåtgången mot antal intervall N i en loglog-plot. Hur beror tidsåtgången av N? (dvs $t \propto N^p$, för vilket p?) Hur är beroendet enligt teorin? Stämmer dina resultat med teorin? (Tips: kolla att det verkligen bara är själva **lösandet** av det linjära ekvationssystemet som ni mäter tid för!)

f) När en matris är gles (fåtal nollskilda element) kan betydligt effektivare metoder än vanlig Gauss-eliminering användas för att lösa ekvationssystemet. Genom att tala om för Matlab att matrisen är gles kommer bättre metoder automatiskt användas när backslash-operatorn anropas. Detta kan ni enkelt göra här genom att skriva `A=sparse(A)`. Gå igenom beräkningarna i deluppgift d igen. (Tiden är nu ännu kortare, kanske behövs fler varv i for-slingan?) Hur stor tidsvinst gör man i detta fall genom att låta Matlab använda metoder för glesa matriser? Plotta tidsåtgången mot antal delintervall N i en loglog-plot. Hur beror tidsåtgången av N? (dvs $t \propto N^p$, för vilket p?) Hur är beroendet enligt teorin för glesa matriser? Stämmer dina resultat med teorin?

Tre små kommentarer:

- Kan det vara så att ens resultat inte riktigt tycks stämma med teorin i boken? Hur motiverar man sitt (korrekta) resultat då?
- Om man redan från början vet att matrisen skall vara gles kan man istället skapa den med kommandona

  ```
  e = ones(N-1,1)/h^2;
  A = spdiags([e -2*e e], -1:1, N-1, N-1);
  ```

- Faktum är att det egentligen är rätt dumt att dela upp balken i så pyttesmå bitar. Uppdelning i $N = 10$, 20 och 40 följt av ett stegs Richardson-extrapolation ger ett svar med högre noggrannhet trots mindre mängd arbete än körningarna med $N = 1000$ och 2000.
  — Men med så små N får vi så usla tids-studier ;)

Hur många timmar ungefär har den här laborationen tagit?

$/- - -\mathcal{NC} - --/$