KTH Computer Science and Communication

Laboration 1

Ekvationslösning

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!

1. MATLAB-repetition

Repetera dina MATLAB-kunskaper från tidigare, tex. labbarna 6 och 7 i DD1345. Arbeta också/alternativt gärna igenom så många som möjligt av uppgifterna i MÖ-häftet, i första hand no 1-5, 10-12 och 14. Häftet "MATLAB *7 i korthet*" ger en stegvis introduktion till MATLAB som kan vara användbar. Flitigt utnyttjande av MATLABs `help`-kommando rekommenderas också. MATLAB-uppgifterna behöver inte redovisas.

MATLAB *kommer vara vårt verktyg under hela kursen så ni kan spara mycket tid vid de framtida laborationerna genom att göra denna repetition ordentligt.*

2. Olinjär skalär ekvation

Man vill bestämma samtliga rötter till följande skalära ekvation,

$$x - 4\sin(2x) - 3 = 0.$$

Noggrannheten skall vara minst tio korrekta siffror.[1]

a) Rita grafen för $f(x) = x - 4\sin(2x) - 3$ (med MATLAB). Samtliga nollställen till $f(x)$ skall vara med. Hur många rötter finns det?

b) Skriv ett program i MATLAB som beräknar rötterna med fixpunktsiterationen

$$x_{n+1} = -\sin(2x_n) + \frac{5}{4}x_n - \frac{3}{4}.$$

Använd ett avbrottsvillkor för iterationen som ger tio siffrors noggrannhet. Undersök empiriskt vilka av rötterna som kan bestämmas med denna metod. Förklara resultatet teoretiskt.

c) Skriv ett program i MATLAB som beräknar rötterna med Newtons metod och tio siffrors noggrannhet. En tumregel för Newton är att *antalet korrekta siffror dubblas i varje iteration.* Stämmer det?

[1] n siffrors noggrannhet är ekvivalent med att relativfelet är mindre än $\frac{1}{2} \cdot 10^{-n}$.

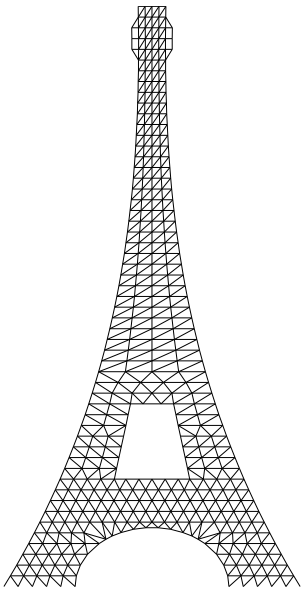
d) Du ska nu jämföra konvergensen för de två metoderna. Välj en rot där bägge metoderna fungerar. Använd samma startgissning. Plotta felet $e_n = |x_n - x|$ efter iteration n som funktion av n för båda metoderna i samma figur. Använd `semilogy`-plot för att få logskala på y-axeln. (För att beräkna felet, gör först en mycket noggrann (15 korrekta siffror) referenslösning med Newtons metod.) Från figuren bör du kunna se tydligt hur mycket snabbare Newton konvergerar än fixpunktiterationer.

Ett annat sätt att kvantifiera hur snabbt metoden konvergerar är *konvergensordningen*. Den beskriver hur mycket mindre felet e_{n+1} är jämfört med e_n. Mer precist är konvergensordningen det största värde p så att e_{n+1}/e_n^p konvergerar mot ett nollskilt tal. (Se sid 23 i Bradie.) Plotta därför e_{n+1}/e_n, e_{n+1}/e_n^2 och e_{n+1}/e_n^3 som funktion av n för de båda metoderna och avgör baserat på detta om metodernas konvergensordning är linjär, kvadratisk eller kubisk.

3. Stora matriser

I många realistiska tillämpningar måste man lösa *stora* linjära ekvationsystem, med tusentals obekanta. Det är i dessa fall som effektiva algoritmer blir viktiga att använda.
Som exempel ska ni här räkna på ett komplicerat fackverk: en modell av Eiffeltornet. Ett fackverk består av stänger förenade genom leder i ett antal noder. Ni ska beräkna deformationen av fackverket när noderna belastas av yttre krafter. Ekvationerna för deformationen härleds i hållfastläran, och baseras på att förskjutningarna i varje nod är små, och att Hookes lag gäller för förlängningen av varje stång.
I slutändan får man ett linjärt ekvationssystem på formen $A\boldsymbol{x} = \boldsymbol{b}$. När antalet noder i fackverket är N kommer antalet obekanta vara $2N$ och $A \in \mathbb{R}^{2N \times 2N}$. Matrisen A brukar kallas *styvhetsmatrisen.* Högerledet $\boldsymbol{b}$ innehåller de givna yttre krafterna som verkar på noderna,

$$\boldsymbol{b} = \Big(F_1^x, F_1^y, F_2^x, F_2^y, \ldots, F_N^x, F_N^y\Big)^T, \qquad \boldsymbol{b} \in \mathbb{R}^{2N},$$



Modellen i `eiffel2.mat`, med 399 noder (798 obekanta).

där $\boldsymbol{F}_j = (F_j^x, F_j^y)^T$ är kraften i nod j. Lösningen $\boldsymbol{x}$ innehåller de resulterande (obekanta) förskjutningarna,

$$\boldsymbol{x} = \Big(\Delta x_1, \Delta y_1, \Delta x_2, \Delta y_2, \ldots, \Delta x_N, \Delta y_N\Big)^T, \qquad \boldsymbol{x} \in \mathbb{R}^{2N}.$$

Här är alltså $(\Delta x_j, \Delta y_j)^T$ förskjutningen av nod j när fackverket belastas med krafterna i $\boldsymbol{b}$.

På kurshemsidan finns filerna `eiffel1.mat`, `eiffel2.mat`, `eiffel3.mat` och `eiffel4.mat`. De innehåller fyra olika modeller av Eiffeltornet med växande detaljrikedom ($N = 261$, 399, 561, 1592). Varje modell består av nodkoordinater i vektorerna `xnod, ynod`, stångindex i matrisen `bars` (används bara för plottningen) och styvhetsmatrisen `A`.

a) Ladda in en av modellerna i MATLAB med kommandot `load`. Hämta funktionsfilen `trussplot.m` från kurshemsidan och anropa den med `trussplot(xnod,ynod,bars)`. för att plotta tornet. Välj nu en av noderna och belasta den med en kraft rakt högerut med beloppet ett. (Sätt $F_j^x = 1$ för

något j, och resten av elementen i $\boldsymbol{b}$ lika med noll, dvs i MATLAB tex: `j=58; b=zeros(2*N,1); b(j*2-1)=1;`) Lös systemet $A\boldsymbol{x} = \boldsymbol{b}$ med backslash för att få fram förskjutningarna i alla punkter. Beräkna de nya koordinaterna för det belastade tornet, $x_j^{\text{bel}} = x_j + \Delta x_j$, etc.:

```
xbel = xnod + x(1:2:end); ybel = ynod + x(2:2:end);
```

Plotta det belastade tornet. Använd `hold on` för att plotta de två tornen ovanpå varandra i samma figur. Markera vilken nod ni valt.

b) Backslash-kommandot i MATLAB använder normalt vanlig gausseliminering för att lösa ekvationsystemet. Undersök hur tidsåtgången för gausseliminering beror på systemmatrisens storlek genom att lösa ekvationsystemet $A\boldsymbol{x} = \boldsymbol{b}$ med ett godtyckligt valt högerled $\boldsymbol{b}$ för var och en av de fyra modellerna. Använd MATLAB-kommandot `cputime`. (`help cputime` ger mer info.) För att få bra noggrannhet i mätningen av CPU-tiden (speciellt om den är kort) bör man upprepa beräkningarna några gånger och ta medelvärdet. Plotta tidsåtgången mot antal obekanta N i en `loglog`-plot. Hur ska tidsåtgången bero på N enligt teorin? Stämmer det?

c) Ni ska räkna ut i vilka noder fackverket är mest respektive minst känslig för horisontell belastning. Börja med den minsta modellen, `eiffel1.mat`. Tag en nod i taget. Belasta den med samma kraft som i a) ovan och räkna ut resulterande förskjutningar $\boldsymbol{x}$. Notera storleken på förskjutningen, dvs $||\boldsymbol{x}||$. Fortsätt med nästa nod, etc. Systematisera beräkningarna med en `for`-slinga i ert MATLAB-program och spara storleken på förskjutningen för varje nod. Ta sedan reda på vilken som nod ger störst respektive minst förskjutning. Plotta tornet med `trussplot` och markera dessa mest och minst känsliga noder i figuren.

Ni kommer att behöva lösa samma stora linjära ekvationssystem med många olika högerled (N stycken). När matrisen är stor, vilket är fallet för de större modellerna, blir detta mycket tidskrävande. Optimera programmet genom att använda LU-faktorisering av A (MATLAB-kommandot `lu`). Uppskatta tidsvinsten av detta. Kan ni köra även de större modellerna?

d) När en matris är gles kan betydligt effektivare metoder än vanlig gausseliminering användas för att lösa ekvationsystemet. Använd `spy(A)` för att studera styvhetsmatrisens struktur. Vad kan man säga om den?

Genom att tala om för MATLAB att matrisen är gles kommer bättre metoder automatiskt användas när backslash anropas. Detta kan ni enkelt göra här genom att skriva `A=sparse(A)`. Gå igenom beräkningarna i c) igen. Hur stor tidsvinst gör man i detta fall genom att låta MATLAB använda metoder för glesa matriser? Prova med och utan LU-faktorisering.

4. Egenvärden

Förskjutningen i förra uppgiften beskrev jämviktsläget när noderna utsattes för yttre krafter. Allmänt kommer kraften på varje nod vid förskjutningen $\boldsymbol{x}$ ges av $\boldsymbol{F}_{\text{nod}} = -A\boldsymbol{x}$. I det dynamiska (tidsberoende) fallet följer $\boldsymbol{x} = \boldsymbol{x}(t)$ Newtons andra lag, vilken därför blir

$$\boldsymbol{F}_{\text{nod}} = m\frac{d^2\boldsymbol{x}}{dt^2} \quad \Rightarrow \quad m\frac{d^2\boldsymbol{x}}{dt^2} + A\boldsymbol{x} = 0,$$

där m är en effektiv massa för noderna (en liten förenkling av verkligheten).

a) Antag att $m = 1$. Visa (med papper och penna) att om λ, $\boldsymbol{y}$ är ett egenvärde respektive en egenvektor till A, så är

$$\boldsymbol{x}(t) = \sin\left(t\sqrt{\lambda}\right)\boldsymbol{y},$$

en lösning till differentialekvationen.

Egenmoderna till A beskriver alltså de möjliga svängningarna i fackverket. Egenvärdet λ motsvarar svängningens frekvens (i kvadrat) och egenvektorn $\boldsymbol{y}$ dess amplitud i varje nod. Den allmänna lösningen till differentialekvationen är en superposition av alla möjliga sådana svängningar:

$$\boldsymbol{x}(t) = \sum_{k=1}^{2N}\left[\alpha_k \sin\left(t\sqrt{\lambda_k}\right) + \beta_k \cos\left(t\sqrt{\lambda_k}\right)\right]\boldsymbol{y}_k,$$

där $\lambda_k, \boldsymbol{y}_k$ är egenvärdena/vektorerna till A och α_k, β_k är konstanter som bestäms av begynnelsedata.

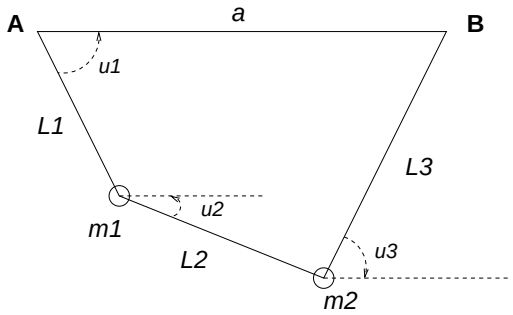
b) Välj en av de mindre modellerna och använd MATLAB-kommandot `eig` för att beräkna de fyra lägsta svängningsmoderna, dvs de fyra minsta egenvärdena och motsvarande egenvektorer. (`eig` returnerar inte egenvärdena i storleksordning. Använd därför `sort`-kommandot på lämpligt sätt för att få fram dem i rätt ordning.) Plotta tornet när noderna är förskjutna med full amplitud och ange frekvensen för var och en av dessa moder. Om inte förskjutningen syns bra, prova att göra egenvektorn längre genom att multiplicera den med ett tal större än ett.

Testa att animera svängningsmoderna med scriptet `trussanim.m`, som finns på kurshemsidan. Prova gärna fler moder.

c) *(Ej obligatorisk uppgift.)* Beräkna det minsta egenvärdet och motsvarande egenvektor med hjälp av inversa potensmetoden (kap 4.2 i Bradie, använd $q = 0$). Hur många iterationer behövs för att få fyra korrekta siffror? Genom att använda LU-faktorisering och metoder för glesa matriser på samma sätt som i förra uppgiften kan man optimera metoden. Kan ni räkna ut det minsta egenvärdet/vektorn även till den största modellen? Skiljer sig dess svängningsmod mycket från den minsta modellen?

5. Ett olinjärt ekvationssystem: Vikter på en lina

Två kulor är fästade på ett snöre som hänger mellan två punkter, A och B. Linjen AB mellan punkterna är horisontell och avståndet mellan punkterna är a. Kulornas massa är m_1 resp m_2. Kulorna delar upp snöret i tre delar med längderna L_1, L_2 och L_3. Uppgiften är att räkna ut vilka vinklar de tre snörena bildar med horisontalplanet samt att rita upp snörets form i en graf.



Beteckna de tre sökta vinklarna u_1, u_2 och u_3. De uppfyller villkoret $\pi/2 > u_1 \geq u_2 \geq u_3 > -\pi/2$ (se figuren). Observera att vridning i motsols riktning ger en positiv vinkel. Rent

geometriskt gäller de två sambanden:

$$L_1 \cos u_1 + L_2 \cos u_2 + L_3 \cos u_3 = a, \qquad L_1 \sin u_1 + L_2 \sin u_2 + L_3 \sin u_3 = 0$$

Dessutom gäller vid jämvikt följande samband:

$$m_2 \tan u_1 - (m_1 + m_2) \tan u_2 + m_1 \tan u_3 = 0$$

De tre sambanden utgör tillsammans ett *icke-linjärt ekvationssystem* för de tre vinklarna u_1, u_2 och u_3. Lös detta ekvationssystem med Newtons metod för följande värden på parametrarna $a = 2$, $L_1 = 1$, $L_2 = 1$ samt L_3, m_1, m_2 enligt tabellen

L_3	m_1	m_2
1	1	1
1	1	2
2	1	1
1	10	1

För varje parameteruppsättning skall, förutom svaret i radianer och grader samt grafen, startgissning och mellanresultat som visar konvergensordningen redovisas.

DN1240, Numeriska metoder gk II

Laboration 1 redovisad och godkänd! Datum: .....

Namn:..... Godkänd av .....