Laboration 2

Laborationen löses i grupper om två och redovisas individuellt genom en lappskrivning den 3/10.

1 Störningsräkning 1

Betrakta funktionen $f(x, y) = e^{yx^2}$. Värdena på x och y är givna av

$$x = 1 \pm 0.01, \qquad y = 2 \pm 0.05.$$

Vi vill uppskatta hur stort fel/osäkerhet vi kommer ha i värdet på f, dvs tex ett svar på formen $f(x, y) = f(1, 2) \pm E_f$. Du ska göra detta på flera olika sätt och jämföra resultaten:

1. Teoretiskt med hjälp av felfortplantningsformeln (Taylor-utveckling).
2. Numeriskt med hjälp av experimentell störningsräkning.
3. Numeriskt med hjälp av min/max-räkning.

2 Störningsräkning 2

Vi vill lösa ekvationen $f(x) = 0$ där $f(x) = \cos(x/50) - 1/\sqrt{2}$. När man gör detta med en numerisk algoritm kan man inte hitta exakt rätt rot x. Istället räknar man fram ett $\tilde{x}$ så att $|f(\tilde{x})| < \varepsilon$ för ett litet ε. Vi vill uppskatta hur stort fel vi kommer ha i approximationen $\tilde{x}$ jämfört med det exakta värdet x när $\varepsilon = 10^{-3}$. Gör det:

1. Teoretiskt med hjälp av felfortplantningsformeln (Taylor-utveckling).
2. Numeriskt genom att undersöka hur mycket du kan störa den exakta roten $x = 25\pi/2$ utan att $|f(x)|$ blir större än 10^{-3}.

3 Icke-linjär 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. 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?
2. Skriv ett program i MATLAB som beräknar rötterna med fixpunktsiterationen

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

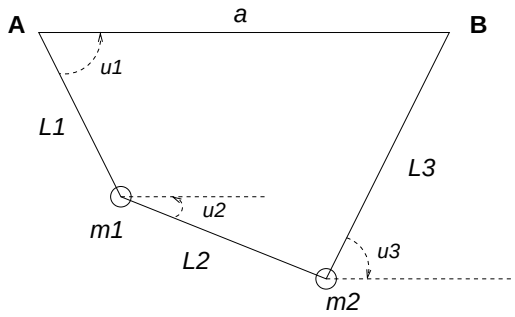
   Använd ett avbrottsvillkor för iterationen som ger tio siffrors noggrannhet. Undersök empiriskt vilka av rötterna som kan bestämmas med denna metod. Förklara resultatet teoretiskt.
3. Skriv ett program i MATLAB som beräknar rötterna med Newtons metod och tio siffrors noggrannhet. En tumregel för Newton är att *antalet korrekta siffror dubblas i varje iteration.* Stämmer det?

4. 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 p om $e_{n+1} \sim e_n^p$ när felen är små. Plotta därför e_{n+1} som en funktion av e_n i `loglog` för de båda metoderna och avgör baserat på detta metodernas konvergensordning.

4 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

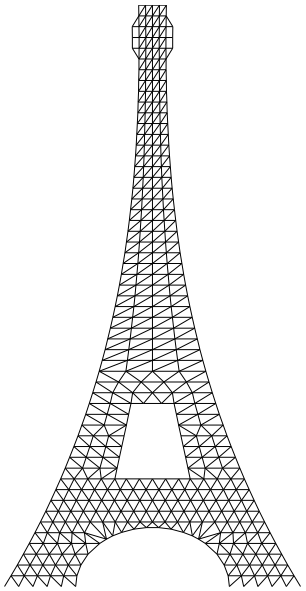
L_3	m_1	m_2
1	1	1
1	1	2
2	1	1
1	10	1

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

5 Linjära ekvationssystem

I många realistiska tillämpningar måste man lösa *stora* linjära ekvationsystem, med miljontals 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 hållfasthetsläran, 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,

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



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,

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

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`.

1. 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=171; 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.

2. 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?

3. 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?

4. 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.

6 Interpolation och minstakvadratanpassning

I almanackan kan man hitta tider för solens upp- och nedgång för några orter i Sverige. Tabellen nedan är uträknad ur almanackan och anger dagens längd i Stockholm den första dagen i varje månad under sommarhalvåret (tiden är angiven decimalt):

Månad:	1 april	1 maj	1 juni	1 juli	1 aug	1 sep
Dagnr :	91	121	152	182	213	244
Solen uppe:	13.18	15.78	17.97	18.38	15.53	14.07

a) Interpolera punkterna med ett femtegradspolynom. Rita de sex givna punkterna och polynomkurvan med fin diskretisering (dagligen från dag 91 till dag 244). Hur länge är solen uppe på nationaldagen den 6 juni? Hur länge är den uppe 15 augusti?

b) Anpassa ett andragradspolynom i minstakvadratmening till den givna datan. Plotta på samma sätt som ovan och ange soltiden 6 juni och 15 augusti enligt denna modell?

c) Tabellen kompletteras med vinterhalvårets värden:

Månad:	1 jan	1 feb	1 mars	1 okt	1 nov	1 dec
Dagnr :	1	32	60	274	305	335
Solen uppe:	6.13	8.02	10.42	11.43	8.73	6.55

Använd det trigonometriska uttryck $c_1 + c_2 \cos \omega t + c_3 \sin \omega t$ där $\omega = 2\pi/365$, som modellfunktion. Varför bör detta kunna ge god anpassning? Rita resultatet i en tvårutors subplot med de tolv punkterna och kurvresultatet (dagligen från dag 1 till dag 365) i första rutan. I andra rutan ritas residualvektorns tolv komponenter mot de tolv givna dagnumren. Beräkna felkvadratsumman samt nationaldagens soltid enligt denna modell.

7 Numerisk derivering och noggrannhetsordning

Betrakta funktionen $f(x) = \sin(x)$.

1. Approximera $f'(1)$ med hjälp av framåtdifferens och (liten) parameter h. Plotta felet i approximationen som funktion av h med `loglog`-kommandot i Matlab. Hur bör felet bero på h i detta fall? Hur kan du använda din plot för att bekräfta detta?

2. För riktigt stora och små värden på h får man dåliga resultat. Vad beror det på? Hur litet fel kan du som bäst uppnå?

3. Gör uppgifterna a-b även för centraldifferenser. Hur ändras resultaten?

4. Läs Sektion 5.1.2 i Sauer och använd detta för att förklara hur stort det minimala felet bör vara teoretiskt för centraldifferensr. Kan du härleda motsvarande formel för framåtdifferenser?