KTH Engineering Sciences

# Laboration 2

## Egenvärden, approximation och numerisk integration

I denna lab ska ni använda numeriska metoder för att beräkna egenvärden, och integraler, samt studera interpolationsmetoder och minstakvadratanpassning. Fokus är på fel och noggrannhetsordning.

Laborationen består av fem uppgifter. För varje uppgift finns en MATLAB-fil i Canvas som utgör en mall för era program. Filerna heter `uppg1.m`, ..., `uppg5.m`. Ni ska programmera de delar som saknas och skicka in de slutgiltiga filerna i Canvas före redovisningen av laborationen.

- De inlämnade uppgifts-filerna ska gå att köra utan några speciella inställningar eller andra hjälpfiler (utöver `trussplot.m`, `trussanim.m` och `dollarkurs.mat` i uppgift 1 och 3). De ska producera de resultat som efterfrågas i respektive uppgift (skriva ut värden, generera plottar, etc.). De behöver inte ge svar på •-frågorna dock.
- I uppgiftsfilerna finns namngivna fördefinierade funktioner som ska implementeras: `potens()`, `inverspotens()`, `trapets()`, `simpson()` och `trapets2d()`. Dessa har färdiga funktionshuvuden givna i filerna. Använd dem! Undvik att ändra syntaxen för funktionerna, speciellt returvärdena. (Ni kan dock lägga till egna funktioner utöver de namngivna om ni vill.) Notera att alla funktioner måste ligga sist i Matlab-filen.
- Gör inga utskrifter eller plottar inne i de namngivna funktionerna. Utskrifter, plottar och annan dataanalys ska göras i huvudprogrammen.
- När flera figurer ska plottas i samma uppgift, använd MATLABs kommando `figure(n)` med olika `n` för att få flera figurfönster.
- Använd två procent-tecken `%%` i början av en rad för att dela in era programfiler i sektioner. Programmet kan då köras sektionsvis med *Run Section* i MATLAB-klienten.
- Använd gärna formaterade utskrifter när ni skriver ut resultat etc. Ni kan tex använda Matlabs `disp()`- eller `fprintf()`-kommandon för detta. Här är exempel på hur de används:

```
>> disp(['a = ' num2str(a) ', och b = ' num2str(b)]);
>> fprintf('Heltal: a = %d och b = %d\n', a, b);
>> fprintf('Flyttal: c = %f och med flera decimaler c=%15.15f\n', c, c);
```

  Med help-kommandot får ni mer information.
- Använd inte MATLABs symboliska funktioner/variabler (`syms`) för beräkningar. Dessa gör programmen betydligt långsammare än vanliga Matlab-funktioner.
- Lämna inte in MATLAB Live Script (`.mlx`-filer). Det introducerar mycket overhead som gör exekvering, och framförallt plottning, långsam.

## 1. Egenvärden

I denna uppgift ska ni beräkna egenvärden och egenvektorer till matriserna för Eiffeltornsmodellerna från Laboration 1. Dessa är nära relaterade till tornets svängningsmoder.

**a) Bakgrund.** Vi såg i Laboration 1 att förskjutningen från jämviktsläget gavs av ekvationen $A\boldsymbol{x} = \boldsymbol{F}$ när $\boldsymbol{F}$ var den yttre kraften på noderna. När systemet inte är jämvikt kommer kraften på varje nod vid förskjutningen $\boldsymbol{x}$ ges av $\boldsymbol{F}_{\text{nod}} = -A\boldsymbol{x}$. (Så att $\boldsymbol{F} + \boldsymbol{F}_{\text{nod}} = 0$ vid jämvikt.) I det tidsberoende fallet beror förskjutningarna på tiden $\boldsymbol{x} = \boldsymbol{x}(t)$, och de följer Newtons andra lag att kraften är lika med massan gånger accelerationen, dvs

$$\boldsymbol{F}_{\text{nod}} = m\frac{d^2\boldsymbol{x}}{dt^2} \quad \Rightarrow \quad m\frac{d^2\boldsymbol{x}}{dt^2} + A\boldsymbol{x} = 0, \tag{1}$$

där $m$ är en effektiv massa för noderna (en liten förenkling av verkligheten) som vi antar är lika med ett, $m = 1$. En lösning till denna ordinära differentialekvation ges av den oscillerande funktionen

$$\boldsymbol{x}(t) = \sin\left(t\sqrt{\lambda}\right)\boldsymbol{y}, \tag{2}$$

där $\lambda$, $\boldsymbol{y}$ är ett egenvärde respektive en egenvektor till $A$-matrisen. Detta gäller för alla par av egenvärden och egenvektorer. Egenmoderna till $A$ beskriver därför möjliga svängningar i fackverket. Egenvärdet $\lambda$ motsvarar svängningens *frekvens* (i kvadrat) och egenvektorn $\boldsymbol{y}$ förskjutningens *amplitud* från jämviktsläget i varje nod.[1]

- Visa att (2) är en lösning till (1). (Med papper och penna.) Ange också mer precist hur frekvensen $f$ ges av egenvärdet, $f = \ldots$. (Detta samband behövs i (b) nedan.)

**b) Visualisering.** För att visualisera egenmoderna, välj den minsta modellen och använd MATLAB-kommandot `eig` för att beräkna egenvärdena och motsvarande egenvektorer. Notera att `eig` inte returnerar egenvärden/egenvektorer i storleksordning. Använd därför `sort`-kommandot på lämpligt sätt för att få dem i rätt ordning. Var noga med att matcha rätt egenvektorer till egenvärdena efter sorteringen.[2]

Använd slutligen `trussplot` for att plotta tornet med noderna förskjutna enligt egenvektorerna. Om egenvektorn är `y` blir plottningen tex

```
>> trussplot(xnod+y(1:2:end), ynod+y(2:2:end), bars);
```

Detta visar egenmodens karaktäristiska form.[3] Plotta de fyra lägsta egenmoderna på detta sätt och ange frekvensen för var och en av dem. De har ganska enkla former. Prova sedan också några högre moder, med mer komplicerade former. Med scriptet `trussanim.m`, som finns i Canvas, kan du slutligen animera svängningsmoderna med förskjutningen $\boldsymbol{x}(t)$ där $t$ är tiden. Syntaxen är

```
>> trussanim(xnod, ynod, bars, y);
```

[1] 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 $\alpha_k, \beta_k$ är konstanter som bestäms av begynnelsedata.

[2] `sort`-kommandot returnerar två argument, där det andra ger en lista på index som är bra att använda.

[3] Om inte förskjutningen syns bra, prova att göra egenvektorn längre genom att multiplicera den med ett tal större än ett.

**c) Beräkning av största och minsta egenvärdena.** För de större modellerna tar `eig`-kommandot ganska lång tid. Ni ska istället använda olika varianter av *potensmetoden* för att beräkna utvalda egenvärden. Fortsättningsvis antas att egenvärdena är ordnade så att

$$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n.$$

Implementera vanliga potensmetoden och inversa potensmetoden i funktionerna `potens()` och `inverspotens()`. Gå igenom de fyra Eiffeltornsmodellerna. Beräkna för var och en av dem det största egenvärdet (med `potens`) och det minsta egenvärdet (med `inverspotens`). Använd toleransen $10^{-10}$. Notera värdena och antal iterationer som krävts. Beräkna även samtliga egenvärden med `eig` och verifiera att era egna metoder räknar rätt. Sammanställ resultaten i en tabell enligt nedan:

| | Största egenvärdet | # iter | $\lambda_2/\lambda_1$ | Minsta egenvärdet | # iter | $\lambda_n/\lambda_{n-1}$ |
|---|---|---|---|---|---|---|
| `eiffel1` | | | | | | |
| `eiffel2` | | | | | | |
| `eiffel3` | | | | | | |
| `eiffel4` | | | | | | |

Optimera era två metoder genom att använda `sparse`-format för matrisen. För `inverspotens()` kan ni också använda LU-faktorisering. Verifiera att era funktioner är snabbare än `eig`. Skriv ut en tidsnotering som visar detta!

- Hur hänger egenvärdeskvoterna i tabellen ihop med antalet iterationer?
- Stämmer teorin i dessa fall?

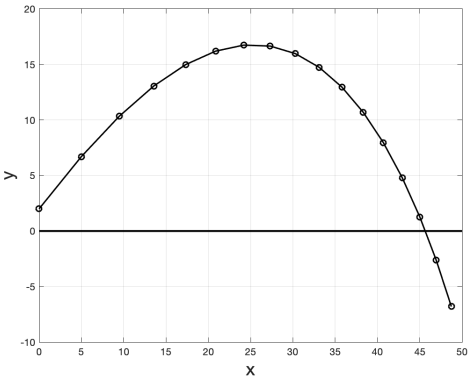
**d) Beräkning av andra egenvärden.** Använd slutligen inversa potensmetoden *med skift* för att hitta de egenvärden till den minsta modellen (`eiffel1.mat`) som ligger närmast 10, 50 och 67. Använd toleransen $10^{-10}$. Skriv ut egenvärdena samt antal iterationer som krävts.

*Ledning:* Anropa `inverspotens()` med $A - \sigma I$ och korrigera det returnerade egenvärdet på lämpligt sätt.

- Vad beror skillnaden i antal iterationer på? Hur hänger det ihop med avstånden mellan egenvärdena? Stämmer teorin?
- Hur skulle man kunna hitta egenvärdet nära 67 snabbare?

## 2. Interpolation

I filen `uppg2.m` finns den givna funktionen `kastbana()` som ger banan för ett kast med en liten boll. Funktionen använder en mycket noggrann numerisk metod för att beräkna banan. Den returnerar bollens position och hastighet vid ett antal ekvidistanta tidpunkter $t_n = nh$. Funktionen anropas som `"[t,x,y,vx,vy]=kastbana(h)"` där `h` är steglängden mellan tidpunkterna och returvärdena är följande vektorer:




- `t` – vektor med tidpunkterna.
- `x, y` – vektorer med bollens koordinater vid tidpunkterna.
- `vx, vy` – vektorer med bollens hastigheter i $x-$ och $y$-led vid tidpunkterna.

Funktionen beräknar banan fram till tiden $t = 5$. En typisk kastbana $y = y(x)$ visas i bilden ovan, där $h = 0.25$ användes. För mer information se kommentarerna i funktionen.

Er uppgift är att lägga till kod i `uppg2.m` som med hjälp av `kastbana()` och styckvis polynominterpolation beräknar tre värden:

- $x_{\text{ned}}$ = $x$-koordinaten för nedslagsplatsen (där $y = 0$).
- $x_{\max}$ = $x$-koordinaten för banans högsta punkt.
- $y_{\max}$ = $y$-koordinaten för banans högsta punkt.

Ert program ska göra detta för ett generellt $h$, med både *linjär* och *kvadratisk* interpolation. Första kommandot i programmet ska tilldela ett värde till variabeln $h$, som inte får ändras senare i programmet. Ni får inte använda MATLABs inbyggda interpolations-funktioner (`polyfit`, `polyval`, `interp1`,...) eller ekvationslösare (`roots`, `fzero`, `fminsearch`, ...) i denna uppgift.[4] Programmet ska skriva ut alla de efterfrågade värdena. Det ska också plotta kastbanan och markera de uträknade max- och nedslagspunkterna i grafen.

Interpolationen kan göras på flera olika sätt (speciellt för högsta punkten). Notera att både punktvärdena $(x, y)$ och hastighetsvärdena $(v_x, v_y)$ kan användas. Oavsett hur interpolationen görs ska följande krav vara uppfyllda när $h = 0.25$. För linjär interpolationen ska felen i $x_{\text{ned}}$ och $y_{\max}$ vara mindre än 0.05, och felet i $x_{\max}$ mindre än 1.2. För kvadratisk interpolationen ska felen i $x_{\text{ned}}$ och $y_{\max}$ vara mindre än 0.005, och felet i $x_{\max}$ mindre än 0.05.

- Prova att använda olika $h$-värden och jämför resultaten för metoderna. Vilken är noggrannast?

*Frivillig utökning:* Implementera också *Hermite-interpolation.* Ni behöver då använda hastighetsvärdena; tänk på att $dy/dx = (dy/dt)/(dx/dt)$. (I utökningen får ni använda det inbyggda kommandot `roots`.)

[4] Använd dem dock gärna för att kolla att er kod räknar rätt!

### 3. Minstakvadratmetoden

På kurshemsidan finns finns datafilen `dollarkurs.mat` som innehåller dagsnoteringar för dollarkursen under två år, från 1a januari 2009 till 31a december 2010. Läs in filen i MATLAB med kommandot `load`. Kursen för dag $i$ betecknas $X_i$, där $i = 1, \ldots, N$ och $N = 730$.

**a)** Anpassa en linjär modell, $f(t) = c_0 + c_1 t$, i minstakvadratmening till dollarkursen genom att ställa upp och lösa lämpligt linjärt ekvationssystem för koefficienterna $c_0$ och $c_1$. Plotta data $X_i$ tillsammans med den anpassade kurvan. Skriv ut värdena på koefficienterna och medelkvadratfelet

$$E = \frac{1}{N}\sum_{i=1}^{N}(X_i - f(i))^2,$$

Plotta även felet mellan den linjära modellen och data.

**b)** Felet i den linjära modellen tycks vara periodiskt. Uppskatta periodlängen $L$ från plotten i förra deluppgiften. Anpassa därefter följande modell till dollarkursen:

$$f(t) = d_0 + d_1 t + d_2 \sin(2\pi t/L) + d_3 \cos(2\pi t/L).$$

Plotta resultatet och felet, som ovan. Skriv ut värdena på koefficienterna och medelkvadratfelet.

**c)** Låt nu även $L$ vara en okänd parameter. Modellen blir då olinjär. Hitta hela den parameteruppsättningen $d_0, d_1, d_2, d_3, L$ som ger bäst approximation av data i minstakvadratmening. Använd Gauss-Newtons metod med resultatet från deluppgift (b) som startgissning. Skriv ut värdena på koefficienterna, inklusive $L$, och medelkvadratfelet.

Plotta slutligen data $X_i$ och alla de tre modellernas anpassning i samma figur.

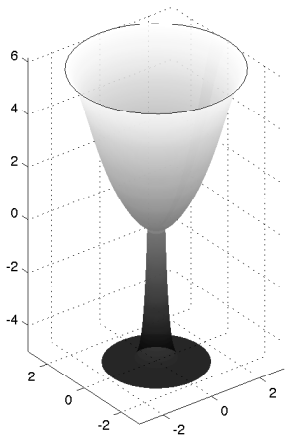
- Hur mycket steg eller föll dollarkursen per dag under perioden enligt den linjära modellen? Ger de andra modellerna väsentligt annorlunda resultat för denna långsiktiga trend?
- Jämför medelkvadratfelen i de olika modellerna. Vilket är störst/minst?

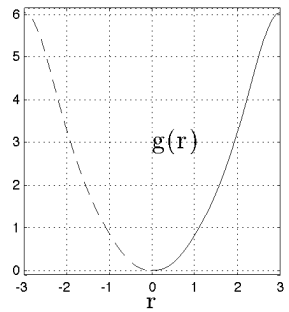
### 4. Numerisk integration

Ett glas har en cirkulär kropp och en öppning med radien $R = 3$. Dess kontur beskrivs av funktionen

$$g(r) = \frac{3r^3 e^{-r}}{1 + \frac{1}{3}\sin\left(\frac{8r}{5}\right)},$$

där $r$ är avståndet till öppningens mitt; se figur. Glasets volym $V$ ges av en dubbelintegral. Med polära koordinater reduceras den till en enkelintegral,






$$V = \int_0^R \int_0^{2\pi} (g(R) - g(r)) r d\theta dr = 2\pi \int_0^R [g(R) - g(r)] r dr = V_0 - 2\pi \int_0^R g(r) r dr.$$

där $V_0 = g(R)R^2\pi$.

**a)** Skriv funktionerna `trapets()` och `simpson()` som beräknar integralen ovan med de sammansatta versionerna av trapetsregeln och Simpsons formel. Funktionerna ska kunna anropas som `"V=trapets(n)"` respektive `"V=simpson(n)"` där `n` är antal delintervall och `V` är approximationen av integralen. Beräkna glasets volym med båda metoderna, för $n = 30$ och $n = 60$, (dvs $h = 0.1$ och $h = 0.05$). Skriv ut resultaten.

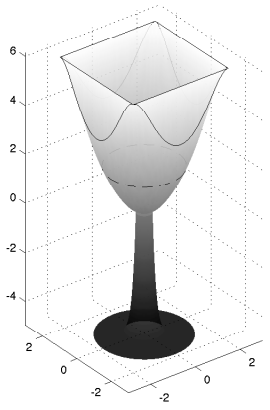
- Hur många decimaler verkar tillförlitliga i resultaten för de två metoderna?
- Vad hade kunnat sägas om tillförlitligheten om man bara beräknat integralen med en steglängd?

**b)** Definiera approximationsfelet $E_h = |V - V_h|$ där $V_h$ är approximationen med steglängd $h$. Använd funktionerna från deluppgift (a) för att mer precist undersöka hur $E_h$ beror på steglängden $h$, på två olika sätt.

1. Gör en mycket noggrann referenslösning $\tilde{V}$ med Simpsons metod[5] och låt $E_h = |\tilde{V} - V_h|$. Plotta sedan $E_h$ som funktion av $h$ för trapetsregeln och Simpsons metod i samma figur. Använd MATLABs kommando `loglog`, gärna tillsammans med `grid`-kommandot. Notera att antal delintervall måste vara jämnt i Simpson-fallet.
   - Uppskatta båda metodernas noggrannhetsordning med hjälp av figuren. Stämmer det med teorin?
2. Bestäm noggrannhetsordningen *utan att använda en referenslösning* genom att beräkna kvoten
$$\frac{V_h - V_{h/2}}{V_{h/2} - V_{h/4}},$$
för en följd av små $h$ och uppskatta noggrannhetsordningen från dessa värden.[6] Skriv ut en tabell med kvoter och motsvarande noggrannhetsordning för trapetsregeln och för Simpsons metod.

**c)** I en alternativ design av glaset har man fasat av sidorna och gjort öppningen kvadratisk med sidan $L = 3\sqrt{2}$; se figur. För att beräkna volymen måste man nu lösa hela dubbelintegralen numeriskt,

$$V = \int_{-L/2}^{L/2} \int_{-L/2}^{L/2} g(R) - g\left(\sqrt{x^2 + y^2}\right)\, dxdy.$$




Implementera trapetsregeln för två dimensioner i funktionen `trapets2d()`, som ska kunna anropas som `"V=trapets2(n)"`, där `n` är antal delintervall i varje koordinatriktning (samma i båda). Beräkna volymen av glaset för $n = 20$ och skriv ut resultatet. Verifiera att noggrannhetsordningen för implementationen är två, genom att plotta felet eller skriva ut kvoterna som i deluppgift (b) ovan.

[5] Eller med MATLABs inbyggda funktion `integral`. Default-noggrannheten i `integral` är dock för låg i detta fall, och måste i så fall ökas.

[6] Se föreläsningsanteckningarna om noggrannhetsordning.

## 5. Högdimensionell numerisk integration

I denna uppgift ska ni beräkna följande tiodimensionella integral på två olika sätt,

$$I = \int_\Omega f(\boldsymbol{x}) d\boldsymbol{x}, \qquad \boldsymbol{x} = (x_1, \ldots, x_{10})^T, \qquad f(\boldsymbol{x}) = e^{x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10}},$$

där $\Omega = [0,\, L]^{10}$ och $L = 1.2$.

**a)** Filen `uppg5.m` innehåller funktionen `trapets10d()` som beräknar integralen med trapetsregeln generaliserad till tio dimensioner. Den anropas som `"I=trapets10d(n)"` där $n$ är antal delintervall i varje koordinatriktning; steglängden är då $h = L/n$. (Studera gärna koden.) Använd funktionen för att beräkna integralvärdet. Ni kommer inte kunna använda speciellt många delintervall $n$ (litet $h$) eftersom beräkningskostnaden ökar mycket snabbt med $n$.

Välj $n$ så att beräkningstiden är mindre än 30 sekunder på er dator och felet så litet som möjligt. Skriv ut fel och beräkningstid.

*Ledning:* Ett bra närmevärde till $I$ ges av $I \approx 6.231467927023725$. (Finns specificerat i `uppg5.m`.)

**b)** Approximera integralen med Monte–Carlo-integration,

$$\int_\Omega f(\boldsymbol{x}) d\boldsymbol{x} \approx I_n = \frac{|\Omega|}{n} \sum_{j=1}^{n} f_j, \qquad f_j = f(\boldsymbol{x}_j), \tag{$*$}$$

där $\boldsymbol{x}_j$ är slumpmässigt valda punkter i $\Omega$ (likformigt fördelade). Generera först $N \times 10$ slumptal med MATLABs `rand`-kommando. Värdet på $N$ ska vara minst $10^6$, men gärna större. Beräkna från dessa slumptal funktionsvärdena $f_1, \ldots, f_N$ och sedan approximationerna $I_1, I_2, \ldots, I_N$, där $I_n$ ges av $(*)$ med hjälp av de första $n$ funktionsvärdena, $f_1, \ldots, f_n$. MATLAB-kommandona `prod` och `cumsum` kommer att vara användbara i dessa steg. Obs! Ni behöver inte generera nya slumptal för varje $I_n$; värdena $f_n$ kan återanvändas när $I_{n'}$ med $n' > n$ beräknas.

Plotta de approximerade integralvärdena $I_n$ som funktion av $n = 1, \ldots, N$ (dvs antal funktionsevalueringar). Plotta också felet som funktion av $n$ i `loglog`-diagram och skriv ut beräkningstiden.

Integralapproximationerna och felet beror på slumptalen. Upprepa därför förfarandet ovan för flera olika följder (minst 5) av slumptal. Detta ger olika *realiseringar* av approximationen. Plotta alla realiseringar tillsammans: en figur för integralapproximationerna och en figur för felen. Använd `axis`-kommandot för att zooma in på den intressanta delen i figuren med integralapproximationerna!

- Baserat på plottarna, uppskatta hur stort $n$ man måste ta för att, med stor sannolikhet, få ett fel mindre än felet ni fick med trapetsregeln. Jämför beräkningstiderna.
- Hur litet fel kan ni få som bäst?
- Överensstämmer felets avtagande med teorin? Motivera! Förklara skillnaden i hur felet beror på antal funktionsevalueringar $n$ för Monte–Carlo och trapetsregeln.

## 6. Frivillig uppgift: Konvergensstudie för styckvis interpolation

Gå tillbaka till uppgift 2 och gör en konvergensstudie för den interpolerade nedslagsplatsen och för maxpunkten. Beräkna först noggrannna referensvärden på dessa genom att välja ett mycket litet $h$. Bestäm sedan felen i de tre kvantiteterna för de två (tre) metoderna för *alla* värden på $h = 5/n$ när $n = 5, 6, 7, \ldots, 200$. Plotta felkurvor och försök förklara resultaten! En del överraskningar bör dyka upp. Vad blir noggrannhetsordningarna? Är felen "snälla" eller inte? Hur skulle felkurvorna se ut om man plottade max-felet i styckvis linjär interpolation?