Laboration 3

3.1 Kvadratur och interpolation

I den här övningen skall vi använda Matlabs `integral`-rutin[1] för att beräkna integraler. Vi skall också öva på att skriva om integralproblem så att de blir numeriskt hanterbara vid svårare fall. Till sist så skall vi se över tre olika metoder för interpolation av mätdata.

3.1.1 Beräkning av integraler med singulär integrand

För hand beräknar vi enkelt att

$$\int_0^1 x^{-7/10}\mathrm{d}x = \frac{10}{3}\left[x^{3/10}\right]_0^1 = \frac{10}{3}.$$

När vi försöker göra detsamma i Matlab får vi däremot ett felmeddelande (det ser ut att i nyare version av Matlab man kan räkna den integral):

```
>> integral(@(x) x.^(-0.7),0,1)
Warning: Infinite or Not-a-Number value encountered.
> In integralCalc/iterateScalarValued (line 349)
  In integralCalc/vadapt (line 132)
  In integralCalc (line 75)
  In integral (line 88)
ans =
   Inf
```

Detta var på grund av att vi naivt delar med noll i den vänstra integrationsgränsen eftersom $x^{-0.7} = 1/x^{0.7}$, for $x = 0$ har vi $1/0$:

```
>> fun = @(x) 1/x.^0.7
fun =
  function_handle with value:
    @(x)1/x.^0.7
>> fun(0)
ans =
   Inf
```

Vi kan rundgå detta på två sätt; genom partiell integration och genom att genomföra ett variabelbyte. Vi skall nu öva på dessa två metoder.

- Beräkna integralen
$$\int_0^1 x^{-9/10}\cos x\mathrm{d}x$$
genom att med hjälp av partiell integration föra över den till en form så att vi slipper dela med noll. Använd sedan `integral`.

- Beräkna samma integral som i (1) genom att göra variabelbytet $x = y^p$ för ett väl valt p. Använd sedan `integral`.

[1] Dokumentation: `https://se.mathworks.com/help/matlab/ref/integral.html`

- (*) Vissa singulariteter kan hanteras av `integral` eftersom de är kända undantagsfall. Ett exempel på detta är

```
>> integral(@(x) log(x),0,1)
ans =
   -1.0000
```

trots att $\log(0) = -\infty$. Med vetskap om detta, beräkna integralen

$$\int_0^1 \frac{\log x}{x^{0.85} + x^{0.9} + x^{0.95}} \mathrm{d}x$$

genom att göra variabelbytet $x = y^p$ för ett väl valt p.

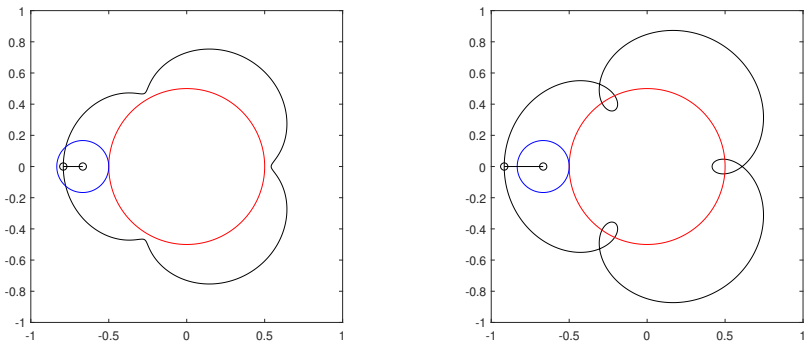
3.1.2 Mer avancerad integration

I den här uppgiften skall vi använda `integral` som ett led i ett större problem.

Antag att vi har tillgång till en maskin som kan skapa bordsskivor som är perfekt formade som en *epicykloid*[2] med tre utbuktningar. En epicykloid med N utbuktningar skapas av att vi låter en liten cirkel med radien $r = R/N$ rulla längs randen på en stor cirkel med radien R. Den parametriseras med avseende på parametern $0 \leq \theta \leq 2\pi$ som

$$\begin{aligned} x(\theta) &= cr\cos(\theta) - qr\cos(c\theta), \\ y(\theta) &= cr\sin(\theta) - qr\sin(c\theta), \end{aligned} \tag{3.1}$$

där $c = (R + r)/r$. I systemet (3.1) q är en parameter som bestämmer hur accentuerade utbuktningarna skall vara, se figuren nedan.



Två exempel på bord som vår bordmaskin kan skapa. Här har vi att $q = 0.75$ respektive $q = 1.5$. Vi har $N = 3$.

Maskinen är inställd på $R = 0.5$ meter, vilket betyder att det enda sättet vi kan ändra på bordets form är att ändra på q. Med allt detta sagt har vi kommit fram till problemet ni skall lösa.

Antag att vi har en tjusig bordslist på 4.25 meter som vi absolut vill sätta på ett epicykloidformat bord.

- För vilket q har vi att omkretsen på bordet är exakt 4.25 meter?

Ledning: Omkretsen kan betraktas som en funktion av q, det vill säga $O(q)$. Omkretsen ges i vanlig ordning av integralen $O(q) = \int_0^{2\pi} \sqrt{(x'(\theta))^2 + (y'(\theta))^2}\mathrm{d}\theta$. Ni kan sedan använda `fsolve` för att lösa problemet $O(q) - 4.25 = 0$.

OBS!: Kom ihåg att ta med alla relevanta beräkningar i er rapport, och inte bara svaret.

[2] Se `https://en.wikipedia.org/wiki/Epicycloid` för mer information och en trevlig animation.

3.1.3 Interpolation av mätdata

I den här uppgiften skall vi anpassa tre typer av interpolationskurvor till mätdata. Datan är given som

```
t = linspace(-2, 3, 8)';
y1 = exp(-t.^2);
y2 = [-1 -1 -1 -1 1 1 1 1]';
```

där `t` är tiderna som mätvärdena `y1`, `y2` uppmättes. I den första dataserien `y1` har ni en underliggande funktion $\exp(-t^2)$ att jämföra med när ni studerar för- och nackdelar med de olika metoderna.

- Ni skall nu använda Matlab-rutinen `interpl`[3] för att anpassa dels en **linjär interpolation**, en **splineinterpolation** och en **shape-preserving interpolation**.
- Plotta alla era tre interpolationer tillsammans med mätdatan i varsin figur för `y1` och `y2`. Avgör vilken interpolationsteknik som matchar datan bäst i båda fallen och motivera era svar.

3.2 Modellering och simulering med ordinära differentialekvationer

I den här uppgiften skall vi simulera en laddad partikel som rör sig i ett elektromagnetiskt fält. Vi antar att vi har N stycken punktladdningar placerade i $\mathbb{R}^2$ på koordinaterna $\mathbf{p}_i$, $i = 1, \ldots, N$. Vi har också en fri partikel som rör sig i $\mathbb{R}^2$ under påverkan av elektromagnetiska krafter från punktladdningarna. Dess position vid tiden t ges av $\mathbf{r}(t) = (x(t), y(t))$, en vektorvärd funktion som uppfyller systemet av differentialekvationer

$$\ddot{\mathbf{r}}(t) = c \sum_{i=1}^{N} \frac{\mathbf{r} - \mathbf{p}}{\|\mathbf{r} - \mathbf{p}\|^3}. \tag{3.2}$$

Här är c är en konstant som tar materialegenskaper (och annat som krävs för att ekvationen skall få rätt enheter) i beaktning.

- Skriv om (3.2) som ett första ordningens system genom att införa hjälpfunktionen $\mathbf{q}(t) = \dot{\mathbf{r}}(t)$. Vi kan sedan skapa en kolonnvektor `y=[r;q]`. $\mathbf{y}(t)$ uppfyller ett första ordningens system av differentialekvationer $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$.
- Skapa en Matlab-funktion `dydt = f(t,y)` som implementerar systemet ni tog fram i föregående uppgift. Som stöd får ni följande kodstubb:

[3] Dokumentation: `https://se.mathworks.com/help/matlab/ref/interpl.html`

```
function dydt = f(t,y)
    global p % Anvand er av globala variabler for punktladdningarnas
    global c % positioner p och konstanten c
    %
    % Skapa hogerledet i systemet. Se till att det fungerar oavsett
    % antalet punktladdningar! Exempelvis genom att lagra punktladd-
    % ningarnas positioner som kolonner i en matris, och sedan lopa
    % igenom denna matris med en for-loop.
    %
    dydt = % skall fyllas i av er
end
```

- Använd `ode45`[4] för att lösa systemet $\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y})$ med hjälp av funktionen ni formulerade i den föregående deluppgiften. Vi antar att det existerar två punktladdningar, en i $p_1 = (-1, -0.9)$ och en i $p_2 = (1, -0.9)$, och att partikeln börjar i vila i positionen $\mathbf{r}(0) = (0.5, -0.9)$. Följande kod kan vara till hjälp:

```
global p
global c
p     = [-1,1;-0.9,-0.9];
c     = 10^(-5);
y0    = [0.5;-0.9;0;0];
t_max = 10^3;
[t,y] = ode45(@f,[0,t_max],y0);
```

- I den föregående uppgiften kommer partikeln att oscillera längs linjen $x = 0.5$ i xy-planet. Om ni alltså får ut att partikelns position är konstant i x-koordinaten är ni på rätt spår. För att göra saker mer intressant så skapar vi en *potentialgrop* som partikeln kan fastna i genom att placera ut N stycken punktkällor länges randen på enhetscirkeln på följande sätt:

```
theta = linspace(0,2*pi,N);
p     = [cos(theta);sin(theta)];
```

- Pröva att variera startposition och starthastighet genom att variera `y0`. Efter att du har tagit fram lösningen kan du animera lösningen med hjälp av följande kodsnutt:

[4] Dokumentation: `https://se.mathworks.com/help/matlab/ref/ode45.html`.

```
for i=1:size(y,1)
    clf
    plot(p(1,:),p(2,:),'r*')
    hold on
    plot(y(i,1),y(i,2),'bo')
    axis([-2.5 2.5 -2.5 2.5])
    axis equal
    drawnow;
    pause(1/60) % 60FPS
end
```

- Lek gärna runt med olika kombinationer av begynnelsevillkor, samt andra konfigurationer på punktladdningarna.
- Till själva inlämningen vill vi att ni skickar med minst två exempel på hur partikeln rör sig.
- Ni kan spara animation som ni har genererat ovan i `gif`-format med hjälp av följande kod.

```
h = figure;
axis tight manual
filename = 'laddad partikel.gif';
for i = 1:size(y,1)
clf
plot(p(1,:),p(2,:),'r*')
hold on
plot(y(i,1),y(i,2),'bo')
axis([2.5 2.5 2.5 2.5])
axis equal
drawnow;
frame = getframe(h);
im = frame2im(frame);
[imind,cm] = rgb2ind(im,256);
if i == 1
imwrite(imind,cm,filename,'gif', 'Loopcount',inf);
else
imwrite(imind,cm,filename,'gif','WriteMode','append');
end
end
```

Inlämning

Skicka era svar på de samtliga frågor markerade i punktlistorna ovan som en pdf via Canvas, se deadline i CANVAS. Glöm inte att bifoga väl kommenterad kod till varje uppgift. Koden skall bifogas som en enda m-fil vid namn **namn1_namn2_MMG410Lab3.m**, och varje uppgift skall uppta en egen cell i filen. Bifoga även `gif`-animationerna tillhörande uppgift 3.2.