Datorlaboration VT22

Mål

Målet med laborationen är att använda Matlab för att visualisera tredimensionella objekt, numeriskt beräkna partiella derivator och att söka extrempunkter med hjälp av olika numeriska metoder. Man bör ha läst **kapitel 14.7 i Calculus** om Newtons metod innan man påbörjar denna laboration.

Individuella lösningar

Laborationen utförs och redovisas i grupper om 1-2 personer. Ni bildar själva grupper och ansvarar för att alla medlemmar bidrar lika mycket till lösningen av laborationen.

Rapportering

Laborationen kommer att rapporteras via en "uppgift" på Canvas. Varje student lämnar in ett Matlabscript samt en skriftlig rapport i PDF-format. Namnge filerna enligt följande mönster:

`fornamn_efternamn_omlabb2021.m`

`fornamn_efternamn_omlabb2021.pdf`

Notera att rapporten måste lämnas in som PDF och det är **starkt rekomenderat**, men inte obligatoriskt, att använda LaTeX för att skriva rapporten. Om ni ej har använt LaTeX för att skriva rapporter tidigare finns instruktioner på Canvas för hur ni gör det. Där finns även en mall och exempel. Ni kan även att få hjälp med LaTeX under handledningstillfällena.

Handledning

Handledning av laborationen erbjuds via Zoom tisdag 8 mars kl 13.15-16.00 och tisdag 15 mars kl 13.15-16.00. Zoomlänk meddelas via Canvas. Under handledningstiden är två datorsalar i MIT-huset bokade, där finns datorer med Matlab installerat. Ni kan även installera Matlab gratis på egen dator med universitetets campuslicens.

OBS: Börja arbeta med laborationen innan handledningen så att ni kan använda handledningstiden för att ställa relevanta frågor på sådant ni inte kan lösa själva.

Inlämningsdatum

Filerna ska vara inlämnade senast **fredag 18 mars klockan 17:00** via Canvas. Om rapporten inte blir godkänd finns det en möjlighet att lämna in en komplettering fredag 29 april kl 17.00. Om rapporten inte blir godkänd vid kompletteringen finns möjlighet att göra en omlaboration i augusti, eller nästa gång kursen ges.

Instruktioner

1 Matlabscriptet

Tillsammans med den skriftliga rapporten ska ni lämna in ett Matlabscript som vid körning genererar alla figurer och värden som ska finnas med i raporten.

På Canvas finns **en mall för Matlabscriptet**. Mallen heter "namn_namn_flervar2022.m".

Ladda ner filen från Canvas, spara den i en lämplig mapp på datorn och byt namn på filen enligt mönstret som beskrivs under "Rapportering". Alternativt kan ni skapa en script-fil i MATLAB med rätt namn och klistra in hela innehållet från "namn_namn_omlabb2021.m".

OBS: I dessa instruktioner finns ibland föreslag på kod. Denna exempelkod finns redan inlagt som kommentarer i script-mallen, så ni behöver inte klippa och klistra från instruktionerna.

Hur man skriver funktioner i Matlab

Ert Matlabscript kommer att innehålla två typer av funktioner. Vi kan definiera funktionen $f(x) = x^2$ i Matlab med följande syntax:

```
f = @ (x) x.^2
```

Vi kan även definiera mer komplicerade funktioner, till exempel följande funktion som beräknar medelvärdet av komponenterna av en vektor:

```
function A = MyAverage(v)
    n = length(v) ;
    A = 0 ;
    for i = 1 :  n
            A = A + x(i)./n ;
    end
end
```

OBS: Den senare typen av funktioner måste skrivas i slutet av Matlabscriptet.

Viktigt angående matrisoperationer

Matlab står för "Matrix Laboratory" och arbetar bäst med matrisvärda funktioner. Vi vill kunna beräkna $z = f(x, y)$ där x och y är matriser, det innebär att vi beräknar z för alla kombinationer av värden på x och y samtidigt. För att det inte ska bli fel är det därför viktigt att ni använder en punkt framför alla operatorer för att markera att operationerna ska göras komponentvis. Skriv alltså `.*` för multiplikation, `./` för division och `.^` för exponentiering i era funktioner.

2 Funktioner som ska finnas i ert Matlabscript

Ert Matlabscript ska innehålla följande tre funktioner:

- `MyNPderiv` som beräknar en numerisk partiell derivata.
- `MyNewton` som beräknar en numerisk lösning till ett ekvationssystem med hjälp av Newtons metod.
- `MyCritical` som hittar stationära punkter med hjälp av Newtons metod.

2.1 Numerisk partiell derivata

Ert Matlabscript ska innehålla en funktion:

```
function derivative = MyNPderiv(f, a, b, i)
```

Funktionen har som input en funktion $f = f(x, y)$, två tal a, b och ett index $i \in \{1, 2\}$.

Funktionen ska numeriskt beräkna en uppskattning av $f_i(a, b)$, alltså den partiella derivatan av f med avseende på första variabeln om $i = 1$ och med avseende på andra variabeln om $i = 2$.

Testa gärna att er funktion fungerar genom att beräkna partiella derivator för funktioner där ni redan vet svaret, till exempel

```
test1 = MyNPderiv( @(x,y) (x-1).^2 + (y-1).^2, 0, 0, 1)
```

OBS: Under denna laboration ska ni **alltid** använda er egen funktion, `MyNPderiv`, för att beräkna partiella derivator. Det finns inbyggda funktioner i Matlab för att beräkna derivator så väl numeriskt som algebraiskt, ni får inte använda er av dessa!

Tips: Funktionen `MyNPderiv` returnerar ett tal. För att lösa de följande uppgifterna kommer ni ibland att behöva ange en partiell derivata, till exempel $f_1(x, y)$, som en funktion av två variabler. Då kan ni använda syntaxen

```
@ (x,y) MyNPderiv(f, x, y, 1)
```

Tips: Var försiktiga när ni gör numeriska beräkningar. Ett av de vanligaste misstagen är att man använder "små tal" som är så små att datorn avrundar dem till noll. Detta kan leda till konstiga resultat och svårtolkade felmeddelanden!

2.2 Newtons metod

Ert Matlabscript ska innehålla en funktion:

```
function [x,y] = MyNewton(f, g, a, b)
```

Funktionen har som input två funktioner $f = f(x, y)$, $g = g(x, y)$ och två tal a och b.

Funktionen ska använda `MyNPderiv` för att med hjälp av Newtons metod numeriskt beräkna en lösning (x, y) till ekvationssystemet

$$\begin{cases} f(x,y) = 0 \\ g(x,y) = 0 \end{cases}$$

givet en gissning $(x, y) = (a, b)$.

Testa gärna att er funktion fungerar genom att beräkna en lösning till ett ekvationssystem för vilket ni redan vet svaret, till exempel

```
[test2x, test2y] = MyNewton( @(x,y) y+x , @(x,y) y-2 , 0, 0)
```

Tips: Vilket är ekvationssystemet som vi löser i exemplet? Skriv upp ekvationssystemet och lös det för hand, för att försäkra er om att er metod ger rätt svar.

2.3 Stationära punkter

Ert Matlabscript ska innehålla en funktion:

```
function [x,y] = MyCritical(f, a, b)
```

Funktionen har som input en funktion $f = f(x, y)$ och två tal a och b.

Funktionen ska använda `MyNewton` och `MyNPderiv` för att numeriskt beräkna en stationär punkt ("critical point") för funktionen $f(x, y)$ givet en gissning (a, b).

Testa gärna att er funktion fungerar genom att beräkna en stationär punkt för en funktion för vilken ni redan vet svaret, till exempel

```
[test3x, test3y] = MyCritical( @(x,y) (x-1).^2 + (y-1).^2, 0, 0)
```

Tips: Vilken är funktionen som ni bestämmer en extrempunkt för i exemplet? Skriv upp funktionen och bestäm den stationära punkten för hand, för att försäkra er om att er metod ger rätt svar.

3 Ytan av ett berg

I mallen för ert Matlabscript finns en färdig funktion i slutet:

```
function z = topography(x, y)
```

Fuktionen beräknar z-koordinaten för en punkt på ett bergslandsap givet x- och y-koordinaterna. Resten av laborationen kommer att fokusera på att studera denna funktion och att generera olika figurer som representerar bergslandskapet. Alla figurer ska finnas med i den skriftliga rapporten.

3.1 Figur 1

Ert Matlabscript ska generera en 3D-plot av grafen för funktionen `topography` över det kvadratiska området $0 \leq x \leq 10$, $0 \leq y \leq 10$. Använd Matlabs inbygda funktion `surf` för att generera 3D-ploten. Använd till exempel följande kod:

```
xline=linspace(0,10,100); % A vector with x-values from 0 to 10
yline=linspace(0,10,100); % A vector with y-values from 0 to 10
[x, y]=meshgrid(xline,yline);   % Matrices of x and y values
z = topography(x, y);     % Calculates a matrix of z-values
figure(1)
surf(x, y, z, 'EdgeColor', 'none', 'FaceColor', 'interp')
colormap summer
```

Läs gärna dokumentationen för `linspace`, `meshgrid` och `surf`.

`[x, y] = meshgrid(xline, yline)` skapar en matris `x` där varje rad är en kopia av vektorn `xline` och en matris `y` där varje kollon är en kopia av vektorn `yline`.

`surf(x, y, z)` plottar ytan `z=topography(x,y)`. Villkoren `'EdgeColor', 'none'` gör att det inte dras några synliga kanter mellan punkterna. Villkoren `'FaceColor', 'interp'` gör färgerna på ytan slätare.

Tips: Ni kan använda verktyget "Rotate 3D" som finns under menyn "Tools" i föntret för figuren. Använd detta för att få en uppfattning av hur berget ser ut. Hur många toppar och dalar finns det på berget?

3.2 Figur 2

Ert Matlabscript ska plotta nivåkurvor för funktionen `topography` över det kvadratiska området $0 \leq x \leq 10$, $0 \leq y \leq 10$. Använd Matlabs inbygda funktion `contour`. Använd till exempel följande kod:

```
figure(2)
contour(x, y, z, 50, 'g')   % Level curves f(x,y)=C for topography
```

Läs gärna dokumentationen för `contour`.

`contour(x,y,z)` plottar nivåkurvor till funktionen `z=topography(x,y)`. Vilkoret 50 gör att Matlab ritar 50 nivåkurvor, nivåerna väljs automatiskt. Villkoret `'g'` gör att nivåkurvorna är gröna.

Tips: Jämför figur 1 och figur 2. Vad händer på grafen när nivåkurvorna är tätare? Vad händer när nivåkurvorna är glesare? Kan ni se någon annan intressant koppling mellan figurerna?

3.3 Figur 3

Ert Matlabscript ska plotta nivåkurvor för de partiella derivatorna av funktionen `topography` över det kvadratiska området $0 \leq x \leq 10$, $0 \leq y \leq 10$. Speficikt ska ni plotta kurvorna där $f_1(x, y) = 0$ och kurvorna där $f_2(x, y) = 0$.

Använd funktionen `MyNPderiv` för att numeriskt beräkna värden av de partiella derivatorna av `topography` för matriserna `x` och `y`.

Låt tillexempel

```
zx= MyNPderiv(@(x,y) topography(x,y), x, y, 1);
zy= MyNPderiv(@(x,y) topography(x,y), x, y, 2);
```

Och plotta sedan

```
figure(3);
contour(x, y, zx, [0,0], 'r')   % Curves where df/dx=0
hold on
contour(x, y, zy, [0,0], 'p')   % Curves where df/dy=0
hold off
```

Villkoret [0,0] gör att endast nivåkurvorna $f_i(x, y) = 0$ ritas. Villkoren `'r'` och 'p'gör att kurvorna är röda respektive lila.

Tips: Vad händer när de röda och de lila kurvorna skär varandra?

3.4 Figur 4

Ert Matlabscript ska plotta nivåkurvorna för funktionen `topography` och noll-nivåkurvorna för de partiella derivatorna av `topography` i en gemensam figur. Använd till exempel följande kod:

```
figure(4);
contour(x, y, z, 'g')   % Level curves f(x,y)=C for topography
hold on
contour(x, y, zx, [0,0], 'r')   % Curves where df/dx=0
contour(x, y, zy, [0,0], 'p')   % Curves where df/dx=0
hold off
```

Tips: Vilka slutsatser kan man dra från denna figur?

4 Toppar och dalar på berget

Vi kan använda figur 1-4 som uppskatta koordinaterna för topparna och dalarna på berget. För att beräkna mer exakta koordinater kan vi använda numeriska metoder.

4.1 Extrempunkter

Ert Matlabscript ska innehålla en uträkning som använder funktionen `MyCritical` för att bestämma (x, y)-koordinaterna för alla lokala maximi- och minimipunkter för funktionen `topography` inom det kvadratiska området $0 < x < 10$, $0 < y < 10$, givet indada som ni väljer själva.

Använd till exempel följande kod:

```
[MaxX(1),MaxY(1)] = MyCritical(...  , ...  , ...)
[MaxX(2),MaxY(2)] = MyCritical(...  , ...  , ...)
...
[MinX(1),MinY(1)] = MyCritical(...  , ...  , ...)
[MinX(2),MinY(2)] = MyCritical(...  , ...  , ...)
...
```

Ert script ska även beräkna z koordinaterna för den högsta toppen och den djupaste dalen på berget.

Tips: Använd figur 1-4 för att få uppskattningar som kan användas som indata för funktionen `MyCritical`. För att läsa av koordinaterna ur en figur kan ni använda verktyget "Data Tips" som finns i menyn "Tools" i figurfönstret.

OBS: Kom ihåg att en stationär punkt inte nödvändigtvis är en extrempunkt!

4.2 Figur 5

Ert Matlabscript ska innehålla en plott av nivåkurvorna för funktionen `topography` där topparna och dalarna på berget är markerade. Använd till exempel följande kod:

```
figure(5);
contour(x, y, z, 'g')   % Level curves f(x,y)=C for topography
hold on
plot(MaxX, MaxY , 'r*')   % Maximum points
plot(MinX, MinY, 'b*')   % Minimum points
hold off
```

Villkoren `'r*'` och `'b*'` gör att extrempunkterna markeras med röda och blå sjärnor.

Tips: Om det är någon punkt som saknas eller någon punkt som är fel behöver ni justera indatat till `MyCritical`.

5 En bäck på berget - Sjunkgradient

I varje punkt $(a, b, f(a, b))$ på en yta $z = f(x, y)$ ger vektorn $-\nabla f(a, b)$ den brantaste riktningen nedför ytan. Givet en startpunkt startpunkt (x_0, y_0) kan vi rekursivt definiera nya punkter (x_i, y_i) genom

$$x_{i+1} = x_i - af_1(x_i, y_i)$$
$$y_{i+1} = y_i - af_2(x_i, y_i)$$

där a är ett litet (men inte för litet) tal. Vi kan tänka oss att vi går längs ytan i den riktning som lutar brantast nedåt. Detta ger oss en metod att hitta minimipunkter som kan vara effektivare än Newtons metod! Metoden kallas "sjunkgradient" (på engelska "gradient descent") och används bland annat för Machine Learning.

5.1 Figur 5 och 6

Ert Matlabscript ska innehålla en loop som, givet en lämplig startpunkt som ni väljer själva, beräknar tre vektorer `xgdes`, `ygdes` och `zgdes` som ger x, y och z-koordinaterna för de punkter som fås av gradient descent för funktionen `topography`.

OBS: z-koordinaten kan beräknas med hjälp av $z = f(x, y)$.

Ert script ska plotta nivåkurvorna för `topography` tillsammans med kurvan som ges av punkterna (`xgdes`, `ygdes`). Använd till exempel följande kod:

```
figure(6);
contour(x, y, z, 'g')   % Level curves f(x,y)=C for topography
hold on
plot(xgdes, ygdes, 'b')   % A mountain brook!
hold off
```

Ert script ska slutligen generera en 3D-plot av berget och bäcken som rinner ner i dalen. Använd till exempel följande kod:

```
figure(7);
surf(x, y, z, 'EdgeColor', 'none', 'FaceColor', 'interp')
colormap summer
hold on
plot3(xgdes, ygdes, zgdes+0.01, 'b')   % A mountain brook!
hold off
```

Välj startpunkten så att bäcken håller sig inom det kvadratiska området $0 < x < 10$, $0 < y < 10$. Försök att hitta en bäck som rinner från en punkt så nära den **högsta toppen** på berget som möjligt och som slutar i den **djupaste dalen**!

När ni ritar bäckar med olika startpunkter kommer ni också se att det finns några kurvor som bäckarna kommer att söka sig till. Vi kan alltså se hur små bäckar går samman och blidar större bäckar och åar. I botten av dalarna skulle det bildas sjöar.

6 Rapportens innehåll

Den skriftliga rapporten ska innehålla följande:

- En kort introduktion till datorlaborationen, i egna ord. Vad har laborationen gått ut på?
- En kort men korrekt matematisk beskrivning av de metoder som ni använt er av som visar att ni förstår vad ni har gjort:
  - Numerisk partiell derivata
  - Newtons metod
  - Hur man använder Newtons metod för att hitta stationära punkter
  - Sjunkgradient (gradient descent)

  OBS: Ni ska **inte** klistra in Matlabkod i rapporten. Ni ska beskriva de metoder ni använt med hjälp av ord och matematiska formler.
- Alla figurer som genereras av ert Matlabscript:
  - Figur 1
  - Figur 2
  - Figur 3
  - Figur 4
  - Figur 5
  - Figur 6
  - Figur 7

  Var noga med att **ange koordinataxlarna** och att ha en **titel** till varje figur. Använd alternativen som finns i menyn "Insert" i figurfönstren i MATLAB.
- En kort kommentar om varje figur. Vad föreställer figuren och vad säger den om funktionen `topography`? (Se tipsen i instruktionerna.)
- (x, y)-koordinaterna för alla lokala extrempunkter för funktionen `topography` inom området $0 < x < 10$, $0 < y < 10$, beräknat med hjälp av `MyCritical`.

  Presentera punkterna (med fyra decimaler) i en lista med följande format:

x	y	lokalt min/max?
1.0000	4.5000	max
2.2500	3.5000	min
⋮	⋮	⋮

  Förklara kortfattat hur ni vet vilka av punkterna som är lokala minimipunkter och vilka som är maximipunkter. (Kom ihåg att det kan finna stationära punkter som *inte* är extrempunkter. Endast extrempunkter ska vara med i listan!)
- (x, y, z)-koordinaterna för den **högsta toppen** på berget.
- (x, y, z)-koordinaterna för den **djupaste dalen** på berget.
- En avslutande kommentar om vad ni lärt er av laborationen.

OBS: Den skriftliga rapporten ska *inte* innehålla någon MATLAB-kod!