Sidefagssupplering

Geometrisk Design, Læseplan

Kapitel 1

Her læses afsnit 1 til 4, der drejer sig om polynomielle kurver, specielt Bézier kurver. Der skal især lægges vægt på afsnit 4, som giver en fuldstændig gennemgang af teorien, de foregående afsnit kan betragtes som en indledning.

Opgaver:

- 1.3.1.
- 1.3.2, vink: vis først at $\binom{n}{k} = \binom{n-1}{k} + \binom{n-1}{k-1}$.
- 1.3.3.

Øvelser: I Maple® lader vi et punkt i planen være givet som en liste med to tal, f.eks. **p2:=[1, 1]** og vi lader et punkt i rummet være givet som en liste med tre tal, f.eks. **p3:=[1, 0, -1]**.

En Bézier kurve er givet ved sin kontrolpolygon, og i Maple® lader vi en kontrolpolygon og dermed en Bézier kurve være givet ved en liste af punkter, f.eks. **a := [[0,0], [1,1], [0,1], [1,0]];**. Man skal være opmærksom på at Maple® tæller fra 1, så vi har f.eks. **a[1] := [0,0];**. I bogen er det første punkt typisk benævnt $\mathbf{a}_0$. Man kan altså ikke uden videre bruge algoritmerne i bogen; men skal omhyggeligt tænke over hvilke index der skal bruges.

Følgende Maple® kommandoer er bekvemme ved arbejde med lister:

> **a := [[0,0], [1,1], [0,1], [1,0]];**

$$a := [[0, 0], [1, 1], [0, 1], [1, 0]]$$

> **b := [[1,0], [2,2], [3,1], [1,4], [2,2]];**

$$b := [[1, 0], [2, 2], [3, 1], [1, 4], [2, 2]]$$

antal punkter i listen:

> **nops(a), nops(b);**

$$4, 5$$

"Udpakning" af en liste:

> **op(a);**

$$[0, 0], [1, 1], [0, 1], [1, 0]$$

Del af en liste (-1 refererer til sidste element):

```
> b[2..-1];
```

$$[[2, 2], [3, 1], [1, 4], [2, 2]]$$

To lister slås sammen:

```
> [op(a),op(b[2..-1])];
```

$$[[0, 0], [1, 1], [0, 1], [1, 0], [2, 2], [3, 1], [1, 4], [2, 2]]$$

Nedenfor bruges en del kommandoer på formen **plots[***kommando***]**, man kan nøjes med *kommando*, hvis man har brugt kommandoen **with(plots)** først.

- Skriv et Maple®-program, der ved hjælp af rekursionsligningen (1.13) udregner Bernstein polynomierne af en given grad. Der ønskes følgende "skelet" for proceduren *Bernstein*:

```
> Bernstein := proc(n)
>  ...
> end:
> B:=Bernstein(3);
```

$$B := [t \to (1-t)^3, t \to 3(1-t)^2 t, t \to 3(1-t)t^2, t \to t^3]$$

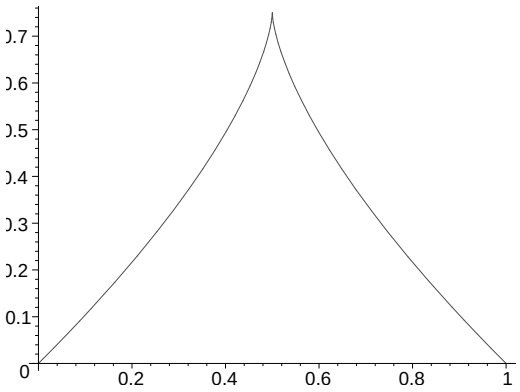
Vi kan nu definerer en Bézier kurve ved

```
> a := [[0,0], [1,1], [0,1], [1,0]]:
> r := expand( sum(B[k](t)*a[k], k=1..4) );
```

$$r := [4t^3 + 3t - 6t^2, -3t^2 + 3t]$$

expand er nødvendig for at få Maple® til at gange ind i de kantede parenteser. Vi bruger Maple®'s **plot**-kommando til at bestemme kurvens graf og Maple®'s **pointplot**-kommando til kontrolpolygonen

```
> curve := plot([r[1], r[2], t=0..1], thickness=3, color=black):
> ctr := plots[pointplot](a, connect=true, thickness=2, linestyle=DASH):
> plots[display]([ctr, curve], scaling=constrained);
```



- Skriv et Maple®-program, der ved hjælp af de Casteljau's algoritme bestemmer et punkt på en Bézier kurve:

```
> deCasteljau := proc(c, t)
>   . . .
> end:
> a := [[0,0], [1,1], [0,1], [1,0]]:
> b := [[0,0,0], [1,0,0], [1,1,0], [1,1,1], [0,1,1]]:
> deCasteljau(a, 0.5);
```

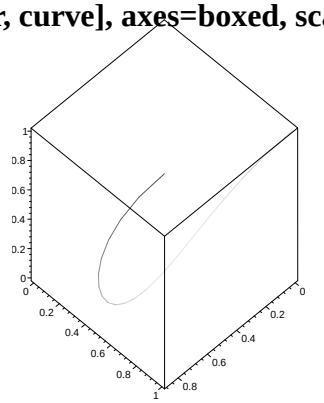
$$[.50, .75]$$

```
> deCasteljau(b, 0.5);
```

$$[.8750, .6875, .3125]$$

Vi kan plotte rum-kurven ved at udregne en række punkter på den og bruge Maple®-kommandoen **pointplot3d** til både kurven og kontrolpolygonen

```
> rr := [seq(deCasteljau(b, i*0.05), i=0..20)]:
> curve := plots[pointplot3d](rr, connect=true, color=black, thickness=3):
> ctr := plots[pointplot3d](b, connect=true, color=black, thickness=2,
>                           linestyle=DASH):
> plots[display]([ctr, curve], axes=boxed, scaling=constrained);
```



- Skriv et Maple®-program der differentierer en Bézier kurve:

```
> BezierDiff := proc(c)
>  ...
> end:
> a := [[0,0], [1,1], [0,1], [1,0]]:
> da := BezierDiff(a);
```

$$da := [[3, 3], [-3, 0], [3, -3]]$$

- Skriv et Maple®-program der øger graden af en Bézier kurve:

```
> DegRaise := proc(c)
>  ...
> end:
> a := [[0,0], [0,4], [4,4], [4,0]]:
> DegRaise(a);
```

$$[[0, 0], [0, 3], [2, 4], [4, 3], [4, 0]]$$

1. Afleveringsopgave

Lav et Maple®-worksheet med en implementering af subdivision ved en given parameterværdi. Input er en kontrolpolygon c for en Bézier kurve $\mathbf{r}$ og en parameterværdi t. Output er et par af kontrolpolygoner $[c_1, c_2]$ for henholdvis $\mathbf{r}_{|[0,t]}$ og $\mathbf{r}_{|[t,1]}$.

```
> subdiv := proc(c, t)
>  ...
> end:
> a := [[0,0], [0,4], [4,4], [4,0]]:
> subdiv(a, 1/2);
```

$$[[0, 0], [0, 2], [1, 3], [2, 3]], [[2, 3], [3, 3], [4, 2], [4, 0]]$$

```
> subdiv(a, 1/4);
```

$$\left[[0, 0], [0, 1], \left[\frac{1}{4}, \frac{7}{4}\right], \left[\frac{5}{8}, \frac{9}{4}\right]\right], \left[\left[\frac{5}{8}, \frac{9}{4}\right], \left[\frac{7}{4}, \frac{15}{4}\right], [4, 3], [4, 0]\right]$$

Brug derefter subdivision ved $\frac{1}{2}$ (eller 0.5) et givet antal gange til at approksimerer en Bézier kurve med den subdividerede kontrolpolygon.

```
> subdiv2 := proc(c, m)
>   . . .
> end:
> subdiv2(a, 0);
```

$$[[0, 0], [0, 4], [4, 4], [4, 0]]$$

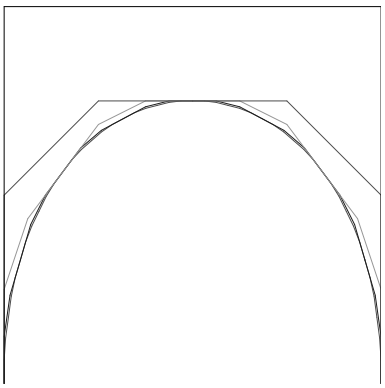
```
> a1 := subdiv2(a, 1);
```

$$a1 := [[0, 0], [0, 2], [1, 3], [2, 3], [3, 3], [4, 2], [4, 0]]$$

```
> a2 := subdiv2(a, 2);
```

$$a2 := \left[[0, 0], [0, 1], \left[\frac{1}{4}, \frac{7}{4}\right], \left[\frac{5}{8}, \frac{9}{4}\right], \left[1, \frac{11}{4}\right], \left[\frac{3}{2}, 3\right], [2, 3], \left[\frac{5}{2}, 3\right], \left[3, \frac{11}{4}\right], \left[\frac{27}{8}, \frac{9}{4}\right], \left[\frac{15}{4}, \frac{7}{4}\right], [4, 1], [4, 0]\right]$$

```
> a3 := subdiv2(a, 3):
> a4 := subdiv2(a, 4):
> p0 := plots[pointplot](a, connect=true, thickness=2, color=black):
> p1 := plots[pointplot](a1, connect=true, thickness=2, color=red):
> p2 := plots[pointplot](a2, connect=true, thickness=2, color=green):
> p3 := plots[pointplot](a3, connect=true, thickness=2, color=blue):
> p4 := plots[pointplot](a4, connect=true, thickness=2, color=red):
> plots[display]([p0, p1, p2, p3, p4], scaling=constrained, axes=none);
```



Kapitel 2

Her læses afsnit 1 til 4, der drejer sig om kurver i planen og rummet.

Opgaver:

- 2.2.11
- 2.2.13
- Lad $\mathbf{r}(t)$ være en regulær kurve med buelængde s, tangentvektor $\mathbf{t}$ og *krumningsvektor* $\boldsymbol{\kappa} = \dfrac{\mathrm{d}\mathbf{t}}{\mathrm{d}s}$. Vis, at Gram-Schmidt ortonormalisrings proceduren anvendt på vektorerne $\mathbf{r}'$ og $\mathbf{r}''$ giver det ortonormale sæt

  $$\mathbf{t},\ \frac{\mathbf{w}}{|\mathbf{w}|}$$

  hvor

  $$\mathbf{w} = \frac{|\mathbf{r}'|^2\mathbf{r}'' - (\mathbf{r}'\cdot\mathbf{r}'')\mathbf{r}'}{|\mathbf{r}'|^2} \quad \text{og} \quad |\mathbf{w}|^2 = \frac{|\mathbf{r}'|^2|\mathbf{r}''|^2 - (\mathbf{r}'\cdot\mathbf{r}'')^2}{|\mathbf{r}'|^2}.$$

  Vis dernæst, ved at benytte $\dfrac{\mathrm{d}}{\mathrm{d}s} = \dfrac{\mathrm{d}t}{\mathrm{d}s}\dfrac{\mathrm{d}}{\mathrm{d}t}$, at

  $$\kappa = \frac{\mathrm{d}\mathbf{t}}{\mathrm{d}s} = \frac{\mathrm{d}^2t}{\mathrm{d}s^2}\mathbf{r}' + \frac{\mathbf{r}''}{|\mathbf{r}'|^2} \quad \text{så} \quad \kappa\cdot\mathbf{w} = \frac{|\mathbf{r}'|^2|\mathbf{r}''|^2 - (\mathbf{r}'\cdot\mathbf{r}'')^2}{|\mathbf{r}'|^4} = \frac{|\mathbf{w}|^2}{|\mathbf{r}'|^2}.$$

  Vis, at $\boldsymbol{\kappa} \perp \mathbf{t}$, så $\boldsymbol{\kappa} = \dfrac{\kappa\cdot\mathbf{w}}{|\mathbf{w}|^2}\mathbf{w} = \dfrac{\mathbf{w}}{|\mathbf{r}'|^2}$, og dermed, at

  $$\boldsymbol{\kappa} = \frac{|\mathbf{r}'|^2\mathbf{r}'' - (\mathbf{r}'\cdot\mathbf{r}'')\mathbf{r}'}{|\mathbf{r}'|^4} \quad \text{og} \quad |\boldsymbol{\kappa}|^2 = \frac{|\mathbf{r}'|^2|\mathbf{r}''|^2 - (\mathbf{r}'\cdot\mathbf{r}'')^2}{|\mathbf{r}'|^6} \qquad (1)$$

- Vis at

  $$\frac{\mathrm{d}|\boldsymbol{\kappa}|^2}{\mathrm{d}t} = \frac{2|\mathbf{r}'|^4(\mathbf{r}''\cdot\mathbf{r}''') - 2|\mathbf{r}'|^2(\mathbf{r}'\cdot\mathbf{r}'')(\mathbf{r}'\cdot\mathbf{r}''') - 6|\mathbf{r}'|^2|\mathbf{r}''|^2(\mathbf{r}'\cdot\mathbf{r}'') + 6(\mathbf{r}'\cdot\mathbf{r}'')^3}{|\mathbf{r}'|^8}$$

  og at

  $$\frac{\mathrm{d}\tau}{\mathrm{d}t} = \frac{[\mathbf{r}', \mathbf{r}'', \mathbf{r}^{(4)}]|\mathbf{r}'\times\mathbf{r}''|^2 - 2[\mathbf{r}', \mathbf{r}'', \mathbf{r}'''](\mathbf{r}'\times\mathbf{r}'')\cdot(\mathbf{r}'\times\mathbf{r}''')}{|\mathbf{r}'\times\mathbf{r}''|^4}$$

- 2.3.5
- 2.3.16
- 2.4.1
- 2.4.3

Øvelser: Bemærk at Maple® både har en **sum**-kommando og en **add**-kommando, og at **sum** kan give uventede resultater:

```
> p := [1, 2, 3]:
> r1 := t -> sum(binomial(2,k)*(1-t)^(2-k)*t^k*p[k+1], k=0..2):
> r2 := t -> add(binomial(2,k)*(1-t)^(2-k)*t^k*p[k+1], k=0..2):
> r1(0),r1(.5),r1(1),r2(0),r2(.5),r2(1);
```

$$0, 2., 0, 1, 2.00, 3$$

I kommandoen **r1** der bruger **sum** bliver 0^k sat lig 0, før k bliver sat til 0.

- Skriv et Maple® program der ved hjælp af numerisk integration (brug **evalf**) bestemmer længden af en Bézier kurve.

```
> Bernstein := proc(n)
>  ...
> end:
> BezierDiff := proc(c)
>  ...
> end:
> BezierLength := proc(c)
>  ...
> end:
> a := [[0,0], [0,1], [1,1], [1,0]]:
> BezierLength(a);
```

$$2.$$

- Skriv et Maple® program der for en Bézier kurve af grad n finder
  1. Længden ℓ_p af kontrolpolygonen.
  2. Afstanden ℓ_c mellem endepunkterne
  3. Det vægtede gennemsnit $(2\ell_c + (n-1)\ell_p)/(n+1)$.

Sammenlign med resultatet af foregående øvelse og undersøg hvad der sker under gentagen subdivision.

```
> BezierLength2 := proc(c)
>   . . .
> end:
> BezierLength2(a);
```

[3., 1., 2.000000000]

```
> subdiv := proc(c, t)
>   . . .
> end:
> BezierLength3 := proc(c, n)
>  local cc;
>  if n=0 then
>    return(BezierLength2(c));
>  else
>    cc:=subdiv(c, .5);
>    return(BezierLength3(cc[1], n-1)+BezierLength3(cc[2], n-1));
>  fi;
> end:
> BezierLength3(a, 0);
```

[3., 1., 2.000000000]

```
> BezierLength3(a, 1);
```

[2.207106782, 1.802775638, 2.004941210]

```
> BezierLength3(a, 2);
```

[2.049793204, 1.950719111, 2.000256158]

```
> BezierLength3(a, 3);
```

[2.012316112, 1.987715841, 2.000015976]

```
> BezierLength3(a, 4);
```

[2.003070730, 1.996931267, 2.000000998]

- Skriv et Maple® program der finder krumningen i et vilkårligt punkt af en plan Bézier kurve.

```
> BezierCurvature2d := proc(c, t)
>   ...
> end:
> BezierCurvature2d(a, 0), BezierCurvature2d(a, .5);
```

$$-.666666667,\ -2.666666667$$

- Skriv et Maple® program der ved hjælp af (1) finder krumningsvektoren i et vilkårligt punkt af en Bézier kurve.

```
> BezierCurvatureVector := proc(c, t)
>   ...
> end:
> b := [[0,0,0], [0,0,1], [0,1,1], [1,1,1], [0,1,1]]:
> BezierCurvatureVector(b, .5);
```

$$\left[\frac{-4}{3},\ \frac{-2}{3},\ \frac{2}{3}\right]$$

- Skriv et Maple® program der ved hjælp af (1) finder længden af krumningsvektoren i et vilkårligt punkt af en Bézier kurve.

```
> BezierCurvature := proc(c, t)
>   ...
> end:
> BezierCurvature(b, .5);
```

$$1.632993162$$

- Skriv et Maple® program der finder torsionen i et vilkårligt punkt af en Bézier kurve i rummet.

```
> BezierTorsion := proc(c, t)
>   ...
> end:
> BezierTorsion(b, 0),BezierTorsion(b, 1/2),BezierTorsion(b, 1);
```

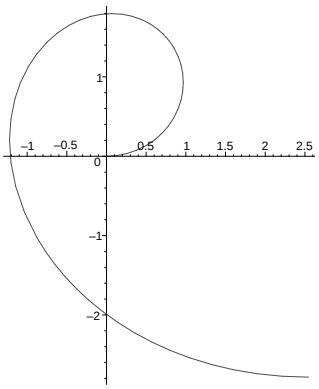
$$\frac{1}{2},\ 0,\ \frac{-1}{2}$$

- Skriv et Maple® program der bestemmer parameterfremstillingen med tangentdrejning for en kurve med den naturlige ligning $\frac{\mathrm{d}s}{\mathrm{d}\varphi} = \varrho(\varphi)$ i det tilfælde hvor $\varrho(\varphi)$ er et polynomium.

```
> PlaneCurve := proc(p, r0, ii)
>  # p en funktion (et polynomium)
>  # r0 startpunktet
>  # ii parameter intervallet
>  . . .
> end:
> p := t->(t/Pi)^3 - (t/Pi)^2 + 1:
> r := PlaneCurve(p, [0,0], 0..2*Pi);
```

$$r := \left[t \to \frac{(3t^2 - 6 - 2\pi t)\cos(t)}{\pi^3} + \frac{(t^3 - 6t - \pi t^2 + 2\pi + \pi^3)\sin(t)}{\pi^3} + \frac{6}{\pi^3}, \right.$$
$$\left. t \to \frac{(6t - t^3 - \pi^3 + \pi t^2 - 2\pi)\cos(t)}{\pi^3} + \frac{(3t^2 - 6 - 2\pi t)\sin(t)}{\pi^3} + \frac{\pi^3 + 2\pi}{\pi^3} \right]$$

```
> plot([op(r), 0..2*Pi], scaling=constrained]);
```



2. Afleveringsopgave

Lav et Maple®-worksheet der kan

1. Plotte krumningen som funktion af buelængden for en vilkårlig Bézier kurve.
2. Plotte torsionen som funktion af buelængden for en vilkårlig Bézier kurve i rummet.

Vi skal altså plotte $|\boldsymbol{\kappa}|$ og τ som funktion af s, dvs. kurverne givet ved $(s, |\boldsymbol{\kappa}(s)|)$ og $(s, \tau(s))$. Disse kan også parametriseres som $(s(t), |\boldsymbol{\kappa}(t)|)$ og $(s(t), \tau(t))$ og det er denne parametrisering vi vil bruge.

Hvis vi har en Bézier kurve $\mathbf{r}(t)$ af grad n, så kan vi tilnærme buelængden $s(t)$ med en kubisk Bézier kurve $\widetilde{s}$ der har kontrol punkter s_0, s_1, s_2, s_3 hvor

$$\begin{aligned} s_3 - s_0 &= \frac{2\ell_c + (n-1)\ell_p}{n+1}, \\ s_1 - s_0 &= \frac{1}{3}s'(0) = \frac{1}{3}|\mathbf{r}'(0)|, \\ s_3 - s_2 &= \frac{1}{3}s'(1) = \frac{1}{3}|\mathbf{r}'(1)|, \end{aligned}$$

bemærk, at s_0 er vilkårlig. Krumningen af $\mathbf{r}$ kan vi tilnærme med en kubisk Bézier kurve $\widetilde{\kappa}$ der har kontrol punkter $\kappa_0, \kappa_1, \kappa_2, \kappa_3$ hvor

$$\kappa_0 = |\boldsymbol{\kappa}(0)|, \quad \kappa_1 - \kappa_0 = \frac{1}{3}\frac{\mathrm{d}|\boldsymbol{\kappa}(0)|}{\mathrm{d}t}, \quad \kappa_3 - \kappa_2 = \frac{1}{3}\frac{\mathrm{d}|\boldsymbol{\kappa}(0)|}{\mathrm{d}t}, \quad \kappa_3 = |\boldsymbol{\kappa}(1)|,$$

bemærk, at $\dfrac{\mathrm{d}|\boldsymbol{\kappa}|}{\mathrm{d}t} = \dfrac{1}{2|\kappa|}\dfrac{\mathrm{d}|\boldsymbol{\kappa}|^2}{\mathrm{d}t}$. Fuldstændigt tilsvarende kan der findes en kubisk tilnærmelse til torsionen. Først implementeres dette. Vi definerer allerførst et par hjælpe kommandoer (prik- og kryds-produkt)

```
> dot := proc(x,y)
>   local k;
>   return(add(x[k]*y[k], k=1..nops(x)));
> end:
> cross := proc(x,y)
>   return(convert(linalg[crossprod](x,y), list));
> end:
> a := [[0,0], [1,1], [0,1], [1,0]]:
> b := [[0,0,0], [0,0,1], [0,1,1], [1,1,1], [0,1,1]]:
> BezierLength1 := proc(c, s0)
```

```
> local s1, s2, s3, ...;
> . . .
> return([s0,s1,s2,s3]);
> end:
> BezierLength1(a, 0); BezierLength1(b, 0);
```

[0, 1.000000000, 1.000000000, 2.000000000]

[0, 1.333333333, 1.632352091, 2.965685424]

```
> BezierCurvature1 := proc(c)
> local k0, k1, k2, k3, ...;
> . . .
> return([k0,k1,k2,k3]);
> end:
> BezierCurvature1(a); BezierCurvature1(b);
```

[.666666666, 1.555555556, 1.555555556, .666666666]

[.750000000, 2.000000000, 2.000000000, .750000000]

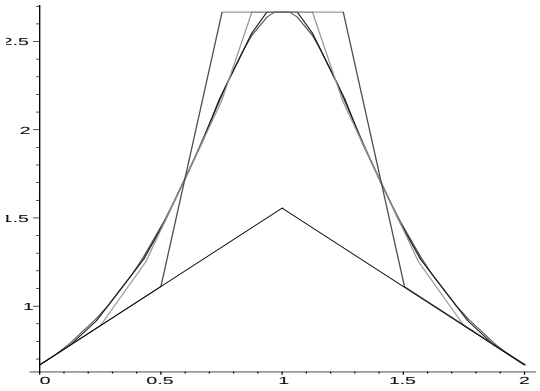
```
> BezierTorsion1 := proc(c)
> local t0, t1, t2, t3, ...;
> . . .
> return([t0,t1,t2,t3]);
> end:
> BezierTorsion1(b);
```

$$\left[\frac{1}{2}, \frac{4}{3}, \frac{-4}{3}, \frac{-1}{2}\right]$$

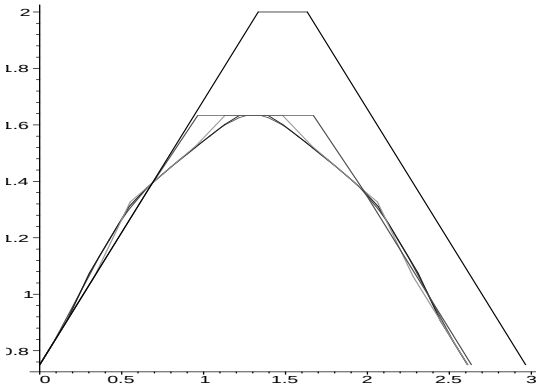
Disse kommandoer kan nu, i lighed med **subdiv2**-kommandoen fra 1. afleverings-opgave, kombineres med subdivision til at give en god tilnærmelse til kurverne $(s(t), |\kappa(t)|)$ og $(s(t), \tau(t))$.

```
> subdiv := proc(c, t)
> . . .
> end:
> BezierCurvaturePlot(c, s0, n)
> . . .
> end:
> p0 := BezierCurvaturePlot(a, 0, 0):
> p1 := BezierCurvaturePlot(a, 0, 1):
> p2 := BezierCurvaturePlot(a, 0, 2):
```

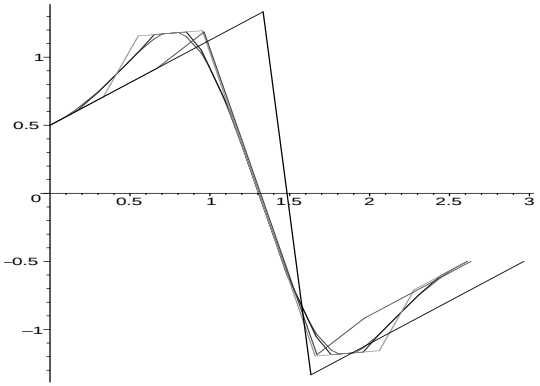
```
> p3 := BezierCurvaturePlot(a, 0, 3):
> p4 := BezierCurvaturePlot(a, 0, 4):
> pp0 := plots[pointplot](p0, connect=true, thickness=2, color=black):
> pp1 := plots[pointplot](p1, connect=true, thickness=2, color=red):
> pp2 := plots[pointplot](p2, connect=true, thickness=2, color=green):
> pp3 := plots[pointplot](p3, connect=true, thickness=2, color=blue):
> pp4 := plots[pointplot](p4, connect=true, thickness=2, color=red):
> plots[display]([pp0,pp1,pp2,pp3,pp4]);
```




```
> p0 := BezierCurvaturePlot(b, 0, 0):
> p1 := BezierCurvaturePlot(b, 0, 1):
> p2 := BezierCurvaturePlot(b, 0, 2):
> p3 := BezierCurvaturePlot(b, 0, 3):
> p4 := BezierCurvaturePlot(b, 0, 4):
> pp0 := plots[pointplot](p0, connect=true, thickness=2, color=black):
> pp1 := plots[pointplot](p1, connect=true, thickness=2, color=red):
> pp2 := plots[pointplot](p2, connect=true, thickness=2, color=green):
> pp3 := plots[pointplot](p3, connect=true, thickness=2, color=blue):
> pp4 := plots[pointplot](p4, connect=true, thickness=2, color=red):
> plots[display]([pp0,pp1,pp2,pp3,pp4]);
```



```
> BezierTorsionPlot(c, s0, n)
>   . . .
> end:
> p0 := BezierTorsionPlot(b, 0, 0):
> p1 := BezierTorsionPlot(b, 0, 1):
> p2 := BezierTorsionPlot(b, 0, 2):
> p3 := BezierTorsionPlot(b, 0, 3):
> p4 := BezierTorsionPlot(b, 0, 4):
> pp0 := plots[pointplot](p0, connect=true, thickness=2, color=black):
> pp1 := plots[pointplot](p1, connect=true, thickness=2, color=red):
> pp2 := plots[pointplot](p2, connect=true, thickness=2, color=green):
> pp3 := plots[pointplot](p3, connect=true, thickness=2, color=blue):
> pp4 := plots[pointplot](p4, connect=true, thickness=2, color=red):
> plots[display]([pp0,pp1,pp2,pp3,pp4]);
```



Kapitel 3

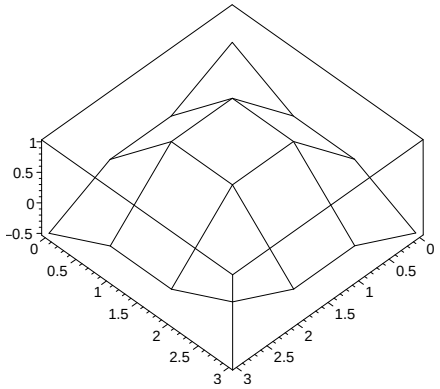
Her læses afsnit 1 til 2, der drejer sig om tensorprodukt Bézier flader.

Opgaver:

- 3.2.4
- 3.2.5

Øvelser: Ligesom vi i kapitel 1 og 2 repræsenterede en Bézier kurve ved en liste af punkter, vil vi her repræsenterer en tensorprodukt Bézier flade ved liste af lister af punkter, f.eks. en bi-kubisk flade:

```
> a := [[[0,0,1/2], [1,0,0], [2,0,0], [3,0,-1/2]],
>       [[0,1,0], [1,1,1], [2,1,1], [3,1,0]],
>       [[0,2,0], [1,2,1], [2,2,1], [3,2,0]],
>       [[0,3,-1/2], [1,3,0], [2,3,0], [3,3,1/2]]]:
> plots[surfdata](a, style=wireframe, color=black, thickness=2,
>             scaling=constrained, axes=boxed);
```

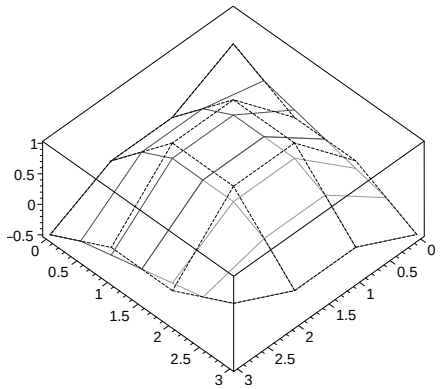


- Skriv et Maple® program der subdividerer en tensor produkt Bézier flade ved en given værdi af den første parameter, og et der subdividerer ved en given værdi af den anden parameter.

```
> subdiv := proc(c, t)
>   # Fra kapitel 1
> end:
> subdivtensor1 := proc(c, u)
>   . . .
> end:
> a1 := subdivtensor1(a, 1/2);
```

$$a1 = \left[\left[\left[\left[0, 0, \frac{1}{2}\right], [1, 0, 0], [2, 0, 0], \left[3, 0, \frac{-1}{2}\right]\right],\right.\right.$$
$$\left[\left[0, \frac{1}{2}, \frac{1}{4}\right], \left[1, \frac{1}{2}, \frac{1}{2}\right], \left[2, \frac{1}{2}, \frac{1}{2}\right], \left[3, \frac{1}{2}, \frac{-1}{4}\right]\right],$$
$$\left[\left[0, 1, \frac{1}{8}\right], \left[1, 1, \frac{3}{4}\right], \left[2, 1, \frac{3}{4}\right], \left[3, 1, \frac{-1}{8}\right]\right],$$
$$\left.\left[\left[0, \frac{3}{2}, 0\right], \left[1, \frac{3}{2}, \frac{3}{4}\right], \left[2, \frac{3}{2}, \frac{3}{4}\right], \left[3, \frac{3}{2}, 0\right]\right]\right]$$
$$\left[\left[\left[0, \frac{3}{2}, 0\right], \left[1, \frac{3}{2}, \frac{3}{4}\right], \left[2, \frac{3}{2}, \frac{3}{4}\right], \left[3, \frac{3}{2}, 0\right]\right],\right.$$
$$\left[\left[0, 2, \frac{-1}{8}\right], \left[1, 2, \frac{3}{4}\right], \left[2, 2, \frac{3}{4}\right], \left[3, 2, \frac{1}{8}\right]\right],$$
$$\left[\left[0, \frac{5}{2}, \frac{-1}{4}\right], \left[1, \frac{5}{2}, \frac{1}{2}\right], \left[2, \frac{5}{2}, \frac{1}{2}\right], \left[3, \frac{5}{2}, \frac{1}{4}\right]\right],$$
$$\left.\left.\left[\left[0, 3, \frac{-1}{2}\right], [1, 3, 0], [2, 3, 0], \left[3, 3, \frac{1}{2}\right]\right]\right]\right]$$

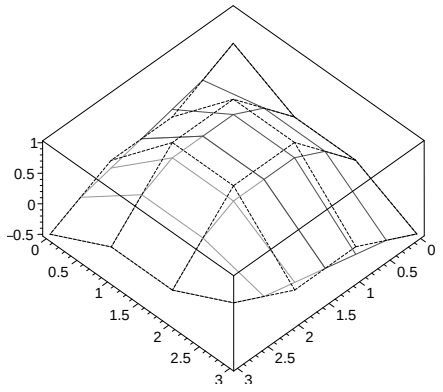
```
> pa := plots[surfdata](a, style=wireframe, color=black, thickness=2,
>       linestyle=DASH):
> p1 := plots[surfdata](a1[1], style=wireframe, color=red, thickness=2):
> p2 := plots[surfdata](a1[2], style=wireframe, color=green, thickness=2):
> plots[display]([pa,p1,p2], scaling=constrained, axes=boxed);
```




```
> subdivtensor2 := proc(c, v)
>   . . .
> end:
> a2 := subdivtensor2(a, 1/2);
```

$$a2 = \left[\left[\left[\left[0, 0, \frac{1}{2}\right], \left[\frac{1}{2}, 0, \frac{1}{4}\right], \left[1, 0, \frac{1}{8}\right], \left[\frac{3}{2}, 0, 0\right]\right],\right.\right.$$
$$\left[[0, 1, 0], \left[\frac{1}{2}, 1, \frac{1}{2}\right], \left[1, 1, \frac{3}{4}\right], \left[\frac{3}{2}, 1, \frac{3}{4}\right]\right],$$
$$\left[[0, 2, 0], \left[\frac{1}{2}, 2, \frac{1}{2}\right], \left[1, 2, \frac{3}{4}\right], \left[\frac{3}{2}, 2, \frac{3}{4}\right]\right],$$
$$\left.\left[\left[0, 3, \frac{-1}{2}\right], \left[\frac{1}{2}, 3, \frac{-1}{4}\right], \left[1, 3, \frac{-1}{8}\right], \left[\frac{3}{2}, 3, 0\right]\right]\right],$$
$$\left[\left[\left[\frac{3}{2}, 0, 0\right], \left[2, 0, \frac{-1}{8}\right], \left[\frac{5}{2}, 0, \frac{-1}{4}\right], \left[3, 0, \frac{-1}{2}\right]\right],\right.$$
$$\left[\left[\frac{3}{2}, 1, \frac{3}{4}\right], \left[2, 1, \frac{3}{4}\right], \left[\frac{5}{2}, 1, \frac{1}{2}\right], [3, 1, 0]\right],$$
$$\left[\left[\frac{3}{2}, 2, \frac{3}{4}\right], \left[2, 2, \frac{3}{4}\right], \left[\frac{5}{2}, 2, \frac{1}{2}\right], [3, 2, 0]\right],$$
$$\left.\left.\left[\left[\frac{3}{2}, 3, 0\right], \left[2, 3, \frac{1}{8}\right], \left[\frac{5}{2}, 3, \frac{1}{4}\right], \left[3, 3, \frac{1}{2}\right]\right]\right]\right]$$

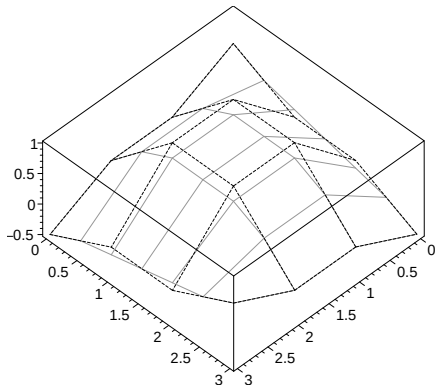
```
> p1 := plots[surfdata](a2[1], style=wireframe, color=red, thickness=2):
> p2 := plots[surfdata](a2[2], style=wireframe, color=green, thickness=2):
> plots[display]([pa,p1,p2], scaling=constrained, axes=boxed);
```



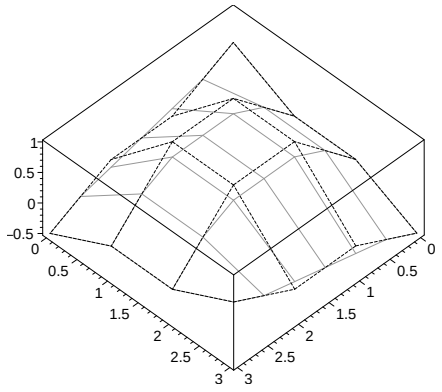
- Skriv et Maple® program som subdividerer en tensor produkt flade *n* gange i den ene retning og *m* gange i den anden retning. I begge tilfælde subdivideres ved parameterværdien $1/2$.

```
> subdiv2 := proc(c, n)
>  # Fra kapitel 1
```

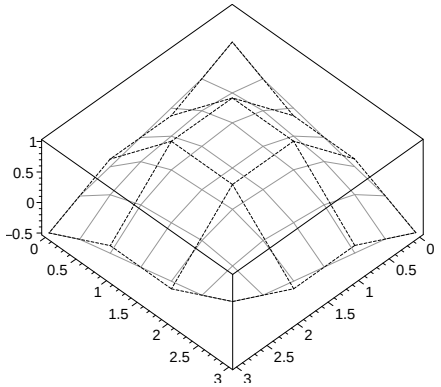
```
> end:
> subdivtensor := proc(c, n, m)
>   . . .
> end:
> aa := subdivtensor(a, 1, 0):
> p := plots[surfdata](aa, style=wireframe, color=green, thickness=2):
> plots[display]([pa,p], scaling=constrained, axes=boxed);
```



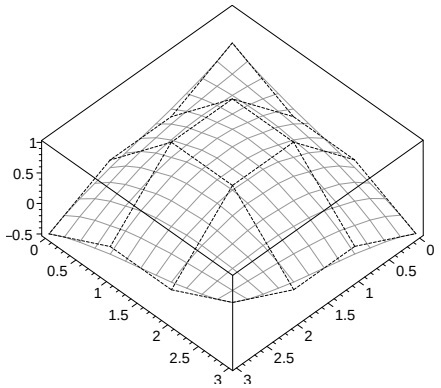

```
> aa := subdivtensor(a, 0, 1):
> p := plots[surfdata](aa, style=wireframe, color=green, thickness=2):
> plots[display]([pa,p], scaling=constrained, axes=boxed);
```



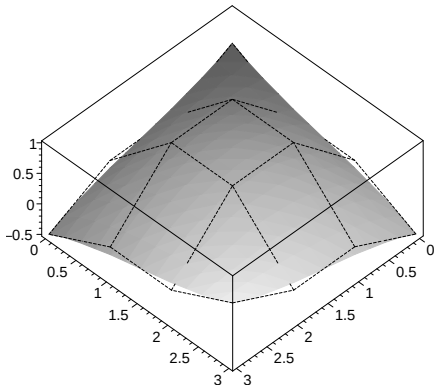

```
> aa := subdivtensor(a, 1, 1):
> p := plots[surfdata](aa, style=wireframe, color=green, thickness=2):
> plots[display]([pa,p], scaling=constrained, axes=boxed);
```




```
> aa := subdivtensor(a, 2, 2):
> p := plots[surfdata](aa, style=wireframe, color=green, thickness=2):
> plots[display]([pa,p], scaling=constrained, axes=boxed);
```




```
> p := plots[surfdata](aa, style=patchnogrid):
> plots[display]([p,pa], scaling=constrained, axes=boxed);
```



- Skriv to Maple® programmer der bestemmer de to partielle afledede af en tensorprodukt Bézier flade.

```
> BezierDiff := proc(c)
>   # Fra kapitel 1
> end:
> TensorBezierDiff1 := proc(c)
>   ...
> end:
> TensorBezierDiff1(a);
```

$$\left[\left[\left[0, 3, \frac{-3}{2}\right], [0, 3, 3], [0, 3, 3], \left[0, 3, \frac{3}{2}\right]\right], [[0, 3, 0], [0, 3, 0], [0, 3, 0], [0, 3, 0]], \left[\left[0, 3, \frac{-3}{2}\right], [0, 3, -3], [0, 3, -3], \left[0, 3, \frac{3}{2}\right]\right]\right]$$

```
> TensorBezierDiff2 := proc(c)
>   ...
> end:
> TensorBezierDiff2(a);
```

$$\left[\left[\left[3, 0, \frac{-3}{2}\right], [3, 0, 0], \left[3, 0, \frac{-3}{2}\right]\right], [[3, 0, 3], [3, 0, 0], [3, 0, -3]], [[3, 0, 3], [3, 0, 0], [3, 0, -3]], \left[\left[3, 0, \frac{3}{2}\right], [3, 0, 0], \left[3, 0, \frac{3}{2}\right]\right]\right]$$

- Skriv to Maple® programmer der hæver graden af en tensorprodukt Bézier flade i hver af de to parametre.

```
> DegRaise := proc(c)
>  # Fra kapitel 1
> end:
> TensorDegRaise1 := proc(c)
>  . . .
> end:
> da := TensorBezierDiff1(a):
> TensorDegRaise1(da);
```

$$\left[\left[\left[0, 3, \frac{-3}{2}\right], [0, 3, 3], [0, 3, 3], \left[0, 3, \frac{3}{2}\right]\right],\right.$$
$$\left[\left[0, 3, \frac{-1}{2}\right], [0, 3, 1], [0, 3, 1], \left[0, 3, \frac{1}{2}\right]\right],$$
$$\left[\left[0, 3, \frac{-1}{2}\right], [0, 3, -1], [0, 3, -1], \left[0, 3, \frac{1}{2}\right]\right],$$
$$\left.\left[\left[0, 3, \frac{-3}{2}\right], [0, 3, -3], [0, 3, -3], \left[0, 3, \frac{3}{2}\right]\right]\right]$$

```
> TensorDegRaise2 := proc(c)
>  . . .
> end:
> da := TensorBezierDiff2(a):
> TensorDegRaise1(da);
```

$$\left[\left[\left[3, 0, \frac{-3}{2}\right], \left[3, 0, \frac{-1}{2}\right], \left[3, 0, \frac{-1}{2}\right], \left[3, 0, \frac{-3}{2}\right]\right],\right.$$
$$[[3, 0, 3], [3, 0, 1], [3, 0, -1], [3, 0, -3]],$$
$$[[3, 0, 3], [3, 0, 1], [3, 0, -1], [3, 0, -3]],$$
$$\left.\left[\left[3, 0, \frac{3}{2}\right], \left[3, 0, \frac{1}{2}\right], \left[3, 0, \frac{1}{2}\right], \left[3, 0, \frac{3}{2}\right]\right]\right]$$

Kapitel 4

Her læses alle afsnit.

Opgaver:

- 4.2.8, 4.2.9 og 4.2.16
- 4.3.3, 4.3.4 og 4.3.6
- 4.4.2, 4.4.3 og 4.4.5

Øvelser:

```
> a := [[[0,0,1/2], [1,0,0], [2,0,0], [3,0,-1/2]],
>       [[0,1,0], [1,1,1], [2,1,1], [3,1,0]],
>       [[0,2,0], [1,2,1], [2,2,1], [3,2,0]],
>       [[0,3,-1/2], [1,3,0], [2,3,0], [3,3,1/2]]]:
> dot := proc(x,y)
>  local k;
>  return(add(x[k]*y[k], k=1..nops(x)));
> end:
> cross := proc(x,y)
>  return(convert(linalg[crossprod](x,y), list));
> end:
```

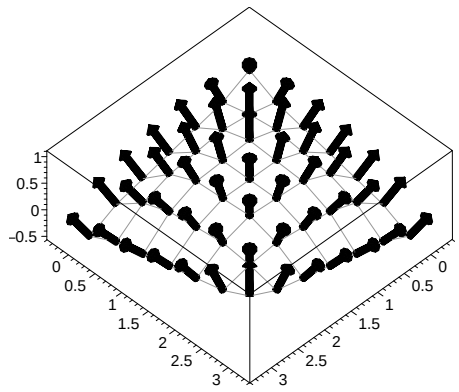
- Skriv et Maple®-program der udregner en approximation til normalvektorfeltet på en tensorprodukt Bézier flade. Opfat, efter subdivision, kontrolnettet for de partielle afledede som en approximation til de partielle afledede og benyt disse værdier til at udregne en approximation til normalvekterfeltet.

```
> subdivtensor := proc(c, n, m)
>  # Fra kapitel 3
> end:
> TensorBezierDiff1 := proc(c)
>  # Fra kapitel 3
> end:
> TensorBezierDiff2 := proc(c)
>  # Fra kapitel 3
> end:
> TensorDegRaise1 := proc(c)
>  # Fra kapitel 3
> end:
```

```
> TensorDegRaise2 := proc(c)
>  # Fra kapitel 3
> end:
>
> TensorNormal := proc(c, n, m)
>  . . .
> end:
> TensorNormal(a, 0, 0);
```

$$
\begin{aligned}
&[[[-0.4082482904, -0.4082482904, -0.8164965812],\\
&[-0.1170411472, 0.7022468831, -0.7022468831],\\
&[-0.1170411472, 0.7022468831, -0.7022468831],\\
&[-0.4082482904, 0.4082482904, -0.8164965812]],\\
&[[0.7022468831, -0.1170411472, -0.7022468831],\\
&[0.3015113445, 0.3015113445, -0.9045340335],\\
&[-0.3015113445, 0.3015113445, -0.9045340335],\\
&[-0.7022468831, 0.1170411472, -0.7022468831]],\\
&[[0.7022468831, -0.1170411472, -0.7022468831],\\
&[0.3015113445, -0.3015113445, -0.9045340335],\\
&[-0.3015113445, -0.3015113445, -0.9045340335],\\
&[-0.7022468831, 0.1170411472, -0.7022468831]],\\
&[[0.4082482904, -0.4082482904, -0.8164965812],\\
&[0.1170411472, -0.7022468831, -0.7022468831],\\
&[0.1170411472, -0.7022468831, -0.7022468831],\\
&[0.4082482904, 0.4082482904, -0.8164965812]]]
\end{aligned}
$$

```
> aa := subdivtensor(a, 1, 1):
> p := plots[surfdata](aa, style=wireframe, color=green):
> N := TensorNormal(a, 1, 1):
> eps:=-.5: # skalering af normalvektor
> Np :=
> plots[display](seq(seq(plottools[arrow](
> aa[i,j],aa[i,j]+eps*N[i,j], .1, .2, .2, cylindrical_arrow),
> i=1..nops(aa)), j=1..nops(aa[1])), color=black):
> plots[display](p, Np, axes=boxed, scaling=constrained);
```



- Skriv et Maple®-program der udregner en approximation til første fundamentalform (E, F og G) på en tensorprodukt Bézier flade. Opfat, efter subdivision, kontrolnettet for de partielle afledede som en approximation til de partielle afledede og benyt disse værdier til at udregne en approximation til første fundamentalform.

```
> TensorEFG := proc(c, n, m)
>   ...
> end:
> TensorEFG(a, 0, 0);
```

$$\left[\left[\left[\frac{45}{4}, \frac{9}{4}, \frac{45}{4}\right], \left[18, \frac{-3}{2}, \frac{37}{4}\right], \left[18, \frac{-3}{2}, \frac{37}{4}\right], \left[\frac{45}{4}, \frac{-9}{4}, \frac{45}{4}\right]\right],\right.$$
$$\left[\left[\frac{37}{4}, \frac{-3}{2}, 18\right], [10, 1, 10], [10, -1, 10], \left[\frac{37}{4}, \frac{-3}{2}, 18\right]\right],$$
$$\left[\left[\frac{37}{4}, \frac{-3}{2}, 18\right], [10, -1, 10], [10, 1, 10], \left[\frac{37}{4}, \frac{-3}{2}, 18\right]\right],$$
$$\left.\left[\left[\frac{45}{4}, \frac{-9}{4}, \frac{45}{4}\right], \left[18, \frac{-3}{2}, \frac{37}{4}\right], \left[18, \frac{-3}{2}, \frac{37}{4}\right], \left[\frac{45}{4}, \frac{9}{4}, \frac{45}{4}\right]\right]\right]$$

- Skriv et Maple®-program der udregner en approximation til anden fundamentalform (L, M og N) på en tensorprodukt Bézier flade. Opfat, efter subdivision, kontrolnettet for de partielle afledede som en approximation til de partielle afledede og benyt disse værdier til at udregne en approximation til anden fundamentalform.

```
> TensorLMN := proc(c, n, m)
>   ...
> end:
> TensorLMN(a, 0, 0);
```

[[[−2.449489744, −11.02270385, −2.449489744],
[4.213481299, −3.160110974, −0.7022468831],
[4.213481299, 1.053370325, 0.7022468831],
[2.449489744, 3.674234615, 2.449489744]],
[[−0.7022468831, −3.160110974, 4.213481299],
[5.427204201, −1.356801050, 5.427204201],
[5.427204201, 0.4522670168, 5.427204201],
[0.7022468831, 1.053370325, 4.213481299]],
[[0.7022468831, 1.053370325, 4.213481299],
[5.427204201, 0.4522670168, 5.427204201],
[5.427204201, −1.356801050, 5.427204201],
[−0.7022468831, −3.160110974, 4.213481299]],
[[2.449489744, 3.674234615, 2.449489744],
[4.213481299, 1.053370325, 0.7022468831],
[4.213481299, −3.160110974, −0.7022468831],
[−2.449489744, −11.02270385, −2.449489744]]]

- Skriv et Maple®-program der udregner en approximation til Gauss krumningen (K) på en tensorprodukt Bézier flade. Opfat, efter subdivision, kontrolnettet for de partielle afledede som en approximation til de partielle afledede og benyt disse værdier til at udregne en approximation til Gauss krumningen.

```
> TensorGaussKrumning := proc(c, n, m)
>  ...
> end:
> TensorGaussKrumning(a, 0, 0);
```

[[−0.9506172856, −0.07881403641, 0.01125914805, −0.06172839509],
[−0.07881403641, 0.2789256197, 0.2954545454, 0.01125914805],
[0.01125914805, 0.2954545454, 0.2789256197, −0.07881403641],
[−0.06172839509, 0.01125914805, −0.07881403641, −0.9506172856]]

- Skriv et Maple®-program der udregner en approximation til Middel krumningen (H) på en tensorprodukt Bézier flade. Opfat, efter subdivision, kon-

trolnettet for de partielle afledede som en approximation til de partielle afledede og benyt disse værdier til at udregne en approximation til Middel krumningen.

```
> TensorMiddelKrumning := proc(c, n, m)
>  ...
> end:
> TensorMiddelKrumning(a, 0, 0);
```

$$[[-1.226804608, 0.01936576980, 0.1503682430, 0.1156934948], \quad [0.01936576980, 0.529607$$

Hvis vi har givet to $n \times m$ matricer, **p** bestående af punkter og **c** bestående af tal, så plotter følgende Maple®-funktion, **MyPlot(a, c)** punkt-matricen som en flade farvelagt efter tal-matricen, hvor det mindste tal svarer til rød og det største tal svarer til violet. Hvis man angiver to extra tal c_1 og c_2 **MyPlot(a, c, c1, c2)** hvor alle tal i **c** ligger i intervallet $[c_1, c_2]$ så kommer c_1 til at svare til rød og c_2 til at scare til violet.

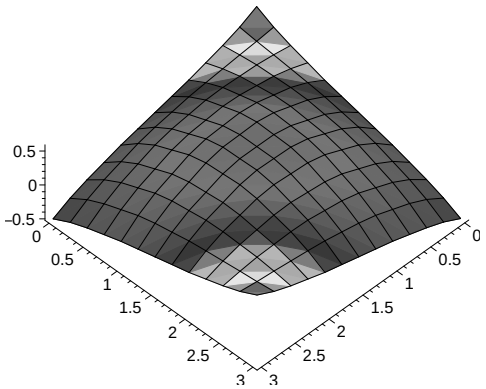
```
> MyPlot := proc(xyz, cc, minc, maxc)
>  local c, c1, c2;
>
>  c  := map(op,cc);
>  if nargs>2 then c1:=minc else c1:=min(op(c)) fi;
>  if nargs>3 then c2:=maxc else c2:=max(op(c)) fi;
>  c := map(x->(x-c1)*.9/(c2-c1),c);
>  PLOT3D( MESH(xyz, COLOR(HUE,op(c))),
>       AXESSTYLE(FRAME), SCALING(CONSTRAINED));
> end:
> aa := subdivtensor(a, 2, 2):
> K := TensorGaussKrumning(a, 2, 2);
> H := TensorMiddelKrumning(a, 2, 2);
> HK := op(map(op,K)),op(map(op,H));
> c1 := min(HK);
```

$$c1 := -1.088433055$$

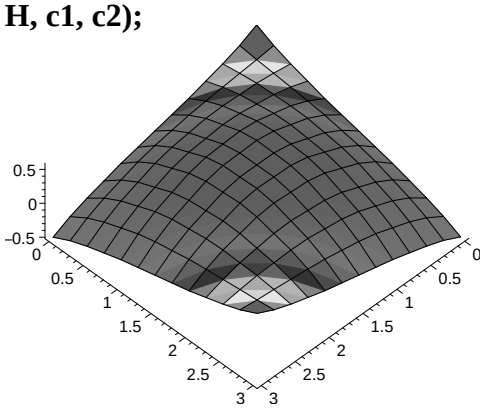
```
> c2 := max(HK);
```

$$c2 := .2343750000$$

```
> MyPlot(aa, K, c1, c2);
```



> **MyPlot(aa, H, c1, c2);**



3. Afleveringsopgave

Betragt en kurve i xz-planen givet ved en parameterfremstilling $I \to \mathbb{R}^2 : u \mapsto \big(r(u), z(u)\big)$ med $r(u) > 0$ for alle $u \in I$. Hvis kurven roteres om z-aksen, får vi en *omdrejningsflade*. Den kan parametrisers som

$$\mathbf{r}(u, v) = \big(r(u)\cos v, r(u)\sin v, z(u)\big) \qquad (*)$$

Antag nu at kurven $\big(r(u), z(u)\big)$ er reulær og injektiv.

1. Vis at $(*)$ er et kort hvis $v \in]v_1, v_2[$ med $v_2 - v1 < 2\pi$.
2. Vis at $M = \big\{\mathbf{r}(u, v) \;\big|\; u \in I, v \in \mathbb{R}\big\}$ er en regulær flade.
3. Find enheds normalvektorfeltet til fladen.
4. Bestem første fundamentalform for fladen.
5. Bestem anden fundamentalform for fladen.
6. Bestem Gauss og middel krumningen for fladen.
7. Bestem de principale krumninger og de principale retninger for fladen.