Flervariabelanalys och MATLAB
Kapitel 3

Thomas Wernstål
Matematiska Vetenskaper

28 september 2012

3 Multipelintegraler

3.1 Dubbelintegraler

I detta kapitel skall vi studera olika sätt på vilket man kan använda MATLAB för att beräkna multipelintegraler, och i detta första avsnitt skall vi speciellt titta på dubbelintegraler. Låt oss börja med ett exempel.

Exempel. Antag att vi vill beräkna dubbelintegralen;

$$\iint_D (y \sin x - x \cos y + 10)\, dxdy$$

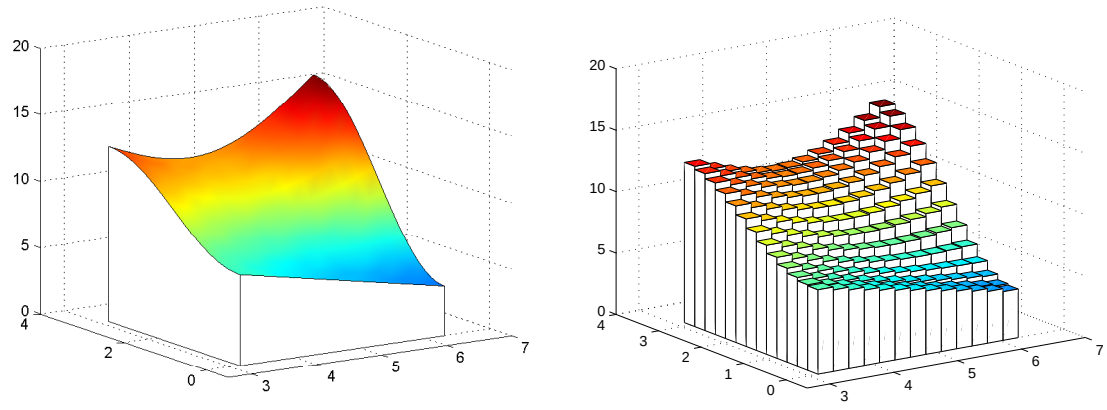
där D är rektangeln $D : \pi \leq x \leq 2\pi\ ,\ 0 \leq y \leq \pi$.

Just denna integral är inte svår att analytiskt beräkna exakt (gör det!) men i allmänhet är det svårt eller omöjligt att beräkna integraler, och vi får nöja oss med närmevärden. Låt oss därför studera några numeriska metoder med vilket integralen kan beräknas approximativt. Ett enkelt sätt att göra detta är genom Riemannsummor;

$$\sum_{i=1}^{n}\sum_{j=1}^{m} f(x_{ij}, y_{ij})\Delta x_i \Delta y_j$$

där x_i och y_j är indelningspunkterna i x- resp. y-led och där $\Delta x_i = x_{i+1} - x_i$, $\Delta y_j = y_{j+1} - y_j$, samt där (x_{ij}, y_{ij}) är en godtycklig punkt i delrektangeln R_{ij} : $x_i \leq x \leq x_{i+1},\ y_j \leq y \leq y_{j+1}$.

Eftersom integranden (i detta fall) är positiv på integrationsområdet kan dubbelintegralens värde tolkas som volymen av en kropp (se figuren till vänster nedan). Varje term i Riemannsumman kan också tolkas som volym, av ett rätblock med basen R_{ij} och höjden $f(x_{ij}, y_{ij})$. Summan av dessa rätblocks-volymer ger en god approximation av dubbelintegralens värde och i gräns, då "indelningens finhet" går mot 0, kommer Riemannsummorna att närma sig integralens exakta värde. Om vi i vårat exempel delar in integrationsområdet i t.ex. 13×13 delrektanglar och väljer $x_{ij} = x_i$ och $y_{ij} = y_j$ (motsvarar det nedre vänstra hörnet i varje delrektangel) så kan Riemannsumman illustreras med figuren till höger nedan.



Om vi vill få en god approximation av dubbelintegralens värde bör vi dock dela in D i många fler delområden. Låt oss därför istället göra en partition av D med 1000×1000 delrektanglar, alla med arean $\pi^2/10^6$;

```
>> n=1000;
>> x=linspace(pi,2*pi,n+1); y=linspace(0,pi,n+1);
>> [X,Y]=meshgrid(x,y);
```

Om vi som ovan väljer $x_{ij} = x_i$ och $y_{ij} = y_j$ så kan motsvarande Riemannsumma beräknas med;

```
>> f=@ (x,y) y.*sin(x)-x.*cos(y)+10;
>> dA=pi^2/n^2;
>> I=sum(sum(f(X(1:n,1:n),Y(1:n,1:n))))*dA
```

Om vi jämför med integralens exakta värde $9\pi^2$ så är det relativa felet av storleksordningen 10^{-4}, vilket får anses ganska stort med tanke på det mycket stora antalet delrektanglar. För att minska relativa felet med en faktor 10, till storleksordningen 10^{-5}, får vi räkna med att öka antalet delrektanglar (och därmed antalet element i matriserna X och Y) med en faktor 100, till 10^8. Det tar lång tid för MATLAB att hantera så stora matriser som med denna metod behövs för att uppnå god noggrannhet, och det är framför allt inte speciellt effektivt. För att ha något att jämföra med när vi lite längre fram skall titta på andra metoder kan vi låta MATLAB mäta den tid kalkylerna tar. Pröva t.ex.

```
>> tic, I=sum(sum(f(X(1:n,1:n),Y(1:n,1:n))))*dA, toc
```

Skall man beräkna Riemannsummor med finare indelning krävs det att kalkylerna organiseras på annat sätt. □

För att beräkna approximativa värden på enkelintegraler (dvs. integraler av funktioner av en variabel) används ofta *Simpsons formel*. Den bygger på att man delar in integrationsområdet i små intervall, säg m stycken, och sedan approximerar integranden i vart och ett av intervallen med ett andragradspolynom. Detta polynom har samma värde som integranden i delintervallets ändpunkter och mittpunkt. Summan av integralerna av andragradspolynomen ger Simpsons formel;

$$\int_a^b f(x)dx \approx \frac{(b-a)}{m} \sum_{k=0}^{2m+1} w_k \cdot f(x_k).$$

Här är $x_k = a + \frac{k}{2m+1}(b-a)$ och w_k är i tur och ordning $\frac{1}{6},\ \frac{4}{6},\ \frac{2}{6},\ \frac{4}{6},\ \frac{2}{6}, \cdots \frac{4}{6},\ \frac{2}{6},\ \frac{4}{6}, \frac{1}{6}$.

Genom upprepad integration, där Simpsons formel används först vid integralberäkning i ena variabeln och sedan i den andra, kan man härleda en tvådimensionell Simpsons formel;

$$\iint_D f(x,y)\,dx\,dy \approx \frac{(b-a)(d-c)}{m^2} \sum_{i=0}^{2m+1} \sum_{j=0}^{2m+1} w_{ij} \cdot f(x_i, y_j).$$

Här är $x_i = a + \frac{i}{2m+1}(b-a)$, $y_j = c + \frac{j}{2m+1}(d-c)$ och $w_{ij} = w_i \cdot w_j$.

forts. Exempel. Låt oss istället använda Simpsons formel för att beräkna dubbelintegralen;

$$\iint_D (y \sin x - x \cos y + 10)\,dxdy$$

där D är rektangeln $D : \pi \le x \le 2\pi\ ,\ 0 \le y \le \pi$.

Med 500 delintervall, och därmed 1001 punkter (inklusive mittpunkterna i delintervallen), får vi följden w_k i MATLAB med;

```
>> m=500; n=2*m+1; w=[1 [3*ones(1,n-2)+(-1).^(2:n-1)] 1]/6;
```

Vikterna w_{ij} samlar vi sedan i matrisen;

```
>> W = w' * w;
```

För att beräkna dubbelintegralen med Simpsons formel, och samtidigt ta beräkningstiden, ger vi sedan följande kommandon;

```
>> x=linspace(pi,2*pi,n); y=linspace(0,pi,n);
>> [X,Y]=meshgrid(x,y);
>> dA=pi^2/m^2;
>> tic, I=sum(sum(W.*f(X,Y)))*dA, toc
```

Tidsåtgången blir ungefär samma som tidigare, men relativa felet blir nu av storleksordningen $1/m^5$. □

MATLAB:s integralberäkningsprogram `quad` (för att beräkna enkelintegraler) använder Simpsons formel men med en adaptiv metod där intervallindelningen styrs av beräkningsresultaten. Där funktionen varierar mycket görs finare indelning, och där variationen är liten görs grov indelning. Det finns också ett par andra beräkningsmetoder att tillgå; `quadl` och `quadgk`. Du kan läsa mer om detta i MATLAB:s *Product Help* under rubriken *Numerical Integration (Quadrature)*

Dubbelintegraler över axelparallella rektanglar kan i MATLAB beräknas med hjälp av kommandot `dblquad`. I princip används upprepad integration genom att MATLAB först gör en indelning av den andra variabelns intervall, beräknar enkelintegralen i alla dessa delningspunkter med avseende på den första variabeln med hjälp av `quad` (om man vill kan man i `dblquad` välja annan metod för beräkning av enkelintegralerna). Simpsons formel tillämpas sedan med dessa beräknade integralvärden. Indelningen av andra variabelns intervall förfinas, integraler beräknas i de nya punkterna, Simpsons formel tillämpas igen. Proceduren upprepas till förändringen är liten nog. Även i detta fall görs indelningen finare där det behövs. Trippelintegralsberäkning, som vi kommer till i avsnitt 3.2 bygger på samma idé.

forts. Exempel. Betrakta åter igen dubbelintegralen;

$$\iint_D (y \sin x - x \cos y + 10)\, dxdy$$

där D är rektangeln $D : \pi \leq x \leq 2\pi\ ,\ 0 \leq y \leq \pi$.

Eftersom vi redan definierat integranden som en anonym funktion kan vi beräkna dubbelintegralen, med relativt fel 10^{-6} vilket är grundinställningen, genom kommandot;

```
>> I = dblquad(f,pi,2*pi,0,pi)
```

Om vi jämför med integralens exakta värde $9\pi^2$ så ser vi att det relativa felet i själva verket är ungefär 10^{-10}. För att minska beräkningstiderna kan vi ibland acceptera större relativt fel. Kommandot;

```
>> I = dblquad(f,pi,2*pi,0,pi,0.006)
```

ger ett större relativa fel än ovan men ändå mycket bättre än vad den första Riemannsumman gav, och dessutom mer än tio gånger så snabbt (kontrollera!). □

Vid beräkning av dubbelintegraler över ett allmännare område D, som ej nödvändigtvis är rektangulärt men som ligger inuti en axelparallell rektangel R, använder vi att;

$$\iint_D f(x,y)\, dxdy = \iint_R f(x,y)\chi_D(x,y)\, dxdy$$

där χ_D är karakteristiska funktionen för området D dvs.

$$\chi(x,y) = \begin{cases} 1 & ,\ \text{då } (x,y) \in D \\ 0 & ,\ \text{då } (x,y) \notin D \end{cases}$$

Exempel. Antag att vi vill beräkna dubbelintegralen;

$$\iint_D (y \sin x - x \cos y + 10)\, dxdy$$

där $D = \{(x, y) \colon 0 \leq x \leq 2\ ,\ 0 \leq y \leq x^2\}$.

Vi definierar då anonyma funktioner som motsvarar den karakteristiska funktionen, integranden, och för enkelhets skull, produkten av dessa två;

```
>> karD = @ (x,y) (0 <= x).* (x <= 2).*(0 <= y).* (y <= x.^2);
>> f = @(x,y) y.*sin(x)-x.*cos(y)+10;
>> fD = @(x,y) karD(x,y).*f(x,y);
```

Eftersom $D \subset \{(x, y) \colon 0 \leq x \leq 2\ ,\ 0 \leq y \leq 4\}$, så beräknar vi sedan integralen med kommandot;

```
>> I = dblquad(fD,0,2,0,4)
```

Varje annan rektangel som omfattar D duger (pröva det!). □

3.2 Trippelintegraler

Det är inte någon väsentlig skillnad på beräkning av trippelintegral och dubbelintegral med MATLAB. Motsvarigheten till `dblquad` för trippelintegraler heter `triplequad`.

Exempel. Låt oss se hur man använder `triplequad` för att beräkna integralen;

$$\iiint_D xy^2z^3\, dxdydz$$

där $D : 0 < x < 1\ ,\ 0 < y < 2\ , 1 < z < 2$.

Vi börjar som tidigare med att definiera integranden som en anonym funktion;

```
>> f=@(x,y,z) x.*y.^2.*z.^3;
```

och för att beräkna integralen ger vi därefter kommandot;

```
>> I=triplequad(f,0,1,0,2,1,2)
```

□

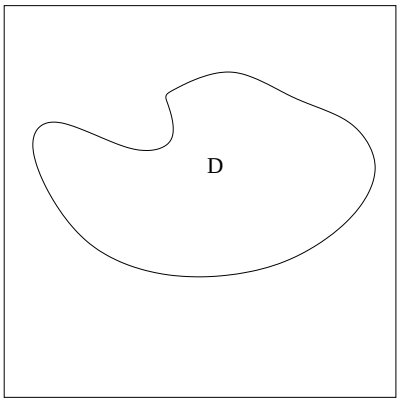
Om integrationsområdet inte är ett rätblock definierar vi en karakteristisk funktion för området, multiplicerar med den och integrerar, precis som i tvådimensionella fallet.

3.3 Monte Carlo metoden

I detta avsnitt skall vi se hur man kan beräkna flerdimensionella integraler approxmativt med den så kallade *Monte Carlo-metoden*. Vi kommer inte att göra någon matematisk analys av hur effektiv och noggrann metoden är, eftersom en sådan analys förutsätter kunskaper i matematisk statistik.

Som nämnts tidigare så använder programmen `quad`, `dblquad` och `triplequad` Simpsons formel på ett adaptivt sätt, där intervallindelningen styrs av beräkningsresultaten. Även om metoden och programmen fungerar ganska bra på de flesta integraler så finns det nackdelar. För det första krävs trots allt ganska många beräkningssteg för att ge önskad noggrannhet. Dessutom ökar beräkningstiden väsentligt med ökande dimension. Vidare har programmen problem med diskontinuiteter, som naturligt kommer in om integrationsområdet inte är en rektangel eller ett rätblock. Det finns heller inget färdigt MATLAB-program för n-dimensionella integraler med $n > 3$.

En enkel metod som kan vara en lösning på dessa problem är *Monte Carlo-metoden*. Låt oss illustrera idén bakom denna metod genom se hur vi kan beräkna arean A av ett område D som ligger inuti en kvadrat med arean 1 (se figur).



Om man väljer en punkt (x, y) slumpvis inuti kvadraten så är sannolikheten att punkten ligger i området D precis A (man måste ställa en del krav på hur området ser ut, för att detta påstående skall vara sant). Om man istället väljer N sådana punkter; (x_i, y_i), $i = 1, 2, ..., N$, så bör alltså i genomsnitt $A \cdot N$ av dessa hamna inuti D. Om M stycken av punkterna ”träffar” D så är alltså $M \approx A \cdot N$, och därmed $A \approx M/N$.

Mer allmänt, om området D istället ligger inom en axelparallell rektangel R med area W så tar man istället punkterna slumpvis i rektangeln R och får att $A \approx W \cdot M/N$.

Med denna metod skall vi alltså;

1. Välja N punkter (x_i, y_i) slumpvis i rektangeln R.
2. Undersöka hur många av de N punkterna som ligger i område D. Kalla detta antal för M.
3. Ta $A \approx W \cdot M/N$ som en approximation av områdets area.

Man kan visa att om området uppfyller vissa villkor (som uppfylls av de flesta områden man kan tänkas komma på) så är beräkningsfelet med denna metod av storleksordning $1/\sqrt{N}$. I det tvådimensionella fallet är detta jämförbart med Riemannsumman men i högre dimension är det väsentligt bättre.

Observera att om χ_D är områdets karakteristiska funktion, och (x_i, y_i) är de slumpvis utvalda punkterna, så ges antalet träffar i D av;

$$M = \sum_{i=1}^{N} \chi_D(x_i, y_i).$$

varpå vi får att;

$$A \approx \frac{W}{N} \sum_{i=1}^{N} \chi_D(x_i, y_i)\,.$$

Exempel. Antag att vi vill beräkna arean av det område D i xy-planet som ges av $1 \leq xy \leq 4,\ x \leq y \leq 4x,\ x > 0,\ y > 0$.

Vi börjar med att skapa den karakteristiska funktionen för området;

```
>> karD=@(x,y) (x.*y>=1).*(x.*y<=4).*(y>=x).*(y<=4*x);
```

De två hyperblerna skär linjerna i punkterna $(\frac{1}{2}, 2), (1, 1), (1, 4), (2, 2)$ och det är inte svårt att se att området D ligger inuti rektangeln $R : \frac{1}{2} \leq x \leq 2,\ 1 \leq y \leq 4$.

Vi behöver sedan generera slumpvisa punkter i rektangeln R. Detta åstadkoms genom att först generera slumpvisa punkter i kvadraten med sida 1, multiplicera med intervallens längder och sedan translatera så att vänster ändpunkt blir rätt;

```
>> N=1000; X=rand(N,2);
>> x=1/2+3/2*X(:,1); y=1+3*X(:,2);
```

Låt oss undersöka vilka punkter detta genererar genom att plotta alla punkter som hamnat inuti D med blå prick och övriga med röd prick;

```
>> in=find(karD(x,y)==1); plot(x(in),y(in),'b.'), hold on
>> out=find(karD(x,y)==0); plot(x(out),y(out),'r.'), hold off
```

Eftersom rektangelarean är $\frac{3}{2} \cdot 3 = \frac{9}{2}$ får vi arean med;

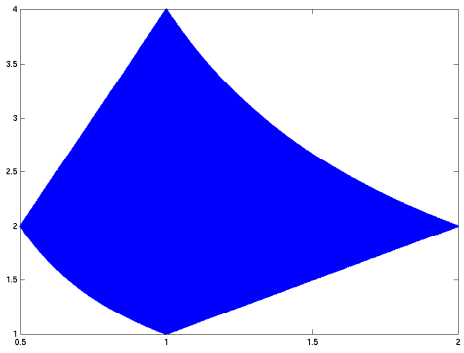
```
>> A=9/2*sum(karD(x,y))/N
```

Nu när vi ser att det hela fungerar kan vi öka antalet punkter t.ex. till en miljon;

```
>> N=10^6; X=rand(N,2);
>> x=1/2+3/2*X(:,1); y=1+3*X(:,2);
>> A=9/2*sum(karD(x,y))/N
```

Detta ger att arean av D är ≈ 2.08, vilket är att jämföra med det exakta värdet som är $3\ln(2)$ (≈ 2.076). En plott med alla punkter som hamnat inuti D ger en fin bild av området D;

```
>> in=find(karD(x,y)==1); plot(x(in),y(in),'.')
```



□

En väsentlig fördel med metoden är att varken beräkningarna eller felet påverkas av dimensionen. Om man har en n-dimensionell kropp som ligger i ett n-dimensionellt rätblock av volym W och slumpmässigt tar N punkter i detta rätblock, så är kroppens (n-dimensionella) volym $V \approx W \cdot \frac{M}{N}$, där M är antalet punkter som ligger i kroppen.

Om man istället vill beräkna en multipelintegral,

$$I \;=\; \int \cdots \int_{\Omega} f(\mathbf{x})\, d\mathbf{x}\,,$$

så gör man precis på samma sätt, fast nu blir

$$I \;\approx\; \frac{W}{N} \sum_{i=1}^{N} \chi_{\Omega}(\mathbf{x}_i) f(\mathbf{x}_i)\,.$$

Det finns förbättringar av metoden som vi inte går närmare in på här men vill du veta mer kan du t.ex. besöka web-sidan;

http://en.wikipedia.org/wiki/Monte_Carlo_method

Exempel. Antag att vi vill beräkna trippelintegralen;

$$\iiint_D 1/\sqrt{x^2+y^2+z^2}\; dxdydz$$

där $D = \{(x,y,z) : x^2+y^2+z^2 \leq 1,\ x>0,\ y>0,\ z>0\}$.

Notera att området D ligger inuti kuben $0 \leq x \leq 1,\ 0 \leq y \leq 1, 0 \leq x \leq 1$.

Som vanligt definierar vi områdets karakteristiska funktion och integranden som anonyma funktioner;

```
>> karD=@(x,y,z) (x.^2+y.^2+z.^2 < 1)
>> f=@(x,y,z) 1./sqrt(x.^2+y.^2+z.^2)
```

och produkten:

```
>> fD=@(x,y,z) karD(x,y,z).*f(x,y,z)
```

Vi låter sedan MATLAB bestämma en miljon punkter i kuben slumpvis med kommandosekvensen;

```
>> N=10^6; X=rand(N,3); x=X(:,1); y=X(:,2); z=X(:,3);
```

Detta ger då integral-approximationen;

```
>> I=sum(fD(x,y,z))/N
```

Det exakta värdet är $\pi/4$ (visa det!) som är ungefär 0.7854, vilket även triplequad ger. Medelvärdet för 10 beräkningar med ovanstående metod blev 0.7858. Styrkan ligger i möjligheten att öka antalet dimensioner. □