Anàlisi Vectorial
Pràctiques amb Maple

Enginyeria de Telecomunicació
Departament de Matemàtica Aplicada IV

març del 2001

Sumari

1 Gràfiques de funcions de dues variables **1**
- 1.1 Introducció . . . 1
- 1.2 Comandes Maple . . . 1
- 1.3 Tutorial . . . 2
  - 1.3.1 Gràfiques de funcions de dues variables . . . 2
  - 1.3.2 Corbes de nivell . . . 3

2 Límits i continuïtat **5**
- 2.1 Introducció . . . 5
- 2.2 Comandes Maple . . . 5
- 2.3 Tutorial . . . 5
  - 2.3.1 Límits direccionals . . . 5
  - 2.3.2 Límits al llarg de corbes . . . 6

3 Diferenciabilitat **8**
- 3.1 Introducció . . . 8
- 3.2 Comandes Maple . . . 8
- 3.3 Tutorial . . . 9
  - 3.3.1 Diferenciabilitat . . . 9
  - 3.3.2 Aproximació lineal en un punt . . . 10

4 Corbes i superfícies **12**
- 4.1 Introducció . . . 12
- 4.2 Comandes Maple . . . 13
  - 4.2.1 Corbes en el pla . . . 13
  - 4.2.2 Superfícies en l'espai . . . 13
  - 4.2.3 Corbes en l'espai . . . 14
- 4.3 Tutorial . . . 14
  - 4.3.1 Corbes al pla . . . 14
  - 4.3.2 Corbes a l'espai . . . 14
  - 4.3.3 Superfícies a l'espai . . . 15
  - 4.3.4 Més exemples . . . 16

5 Polinomis de Taylor **18**
- 5.1 Introducció . . . 18
- 5.2 Comandes Maple . . . 18
- 5.3 Tutorial . . . 18

6 Extrems locals de funcions **20**
- 6.1 Introducció . . . 20
- 6.2 Comandes Maple . . . 20
- 6.3 Tutorial . . . 21
  - 6.3.1 Un exemple . . . 21
  - 6.3.2 Un altre exemple . . . 22

7 Extrems locals condicionats **24**
- 7.1 Introducció . . . 24
- 7.2 Comandes Maple . . . 24
- 7.3 Tutorial . . . 24

8 Integració múltiple **27**
8.1 Introducció . . . 27
8.2 Comandes Maple . . . 27
8.3 Tutorial . . . 28
8.3.1 Un exemple . . . 28
8.3.2 La regla dels rectangles composta . . . 28
8.3.3 La regla de Simpson . . . 29
8.3.4 Un altre exemple . . . 30

9 Integrals de línia i de superfície **31**
9.1 Introducció . . . 31
9.2 Comandes Maple . . . 31
9.3 Tutorial . . . 32

Apèndix: expressions i funcions en Maple **35**

Índex **36**

1 Gràfiques de funcions de dues variables

1.1 Introducció

Una de les aplicacions actuals de la informàtica a la ciència i la tecnologia és la representació gràfica. La visualització d'un problema pot ajudar sovint a resoldre'l de manera més ràpida. En aquesta primera secció veurem com es poden representar gràfiques de funcions de dues variables per tal d'observar-ne algunes característiques rellevants.

Estudiarem funcions $f: U \to \mathbf{R}$, on $U \subset \mathbf{R}^2$ és el *domini* de f, típicament un subconjunt obert.

El *graf* de f és el conjunt de punts

$$\{(x, y, z) \in \mathbf{R}^3 \mid (x, y) \in U, z = f(x, y)\},$$

que és una superfície dins l'espai.

Una *corba de nivell* de f és el conjunt de punts del domini de la funció on aquesta pren un determinat valor constant:

$$C_k = \{(x, y) \in U \mid f(x, y) = k\}.$$

Tot sovint, aquests C_k són corbes dins el domini de f, per això se'n diuen corbes de nivell. Aquests conjunts, per a diferents valors de k, ajuden a visualitzar la gràfica de f. Geomètricament, una corba de nivell es pot obtenir fent la intersecció de la gràfica de f amb un pla horitzontal $z = k$.

1.2 Comandes Maple

La manera de dibuixar la gràfica d'una funció de dues variables amb Maple és:

`plot3d(`*f*`,`*a*`..`*b*`,`*c*`..`*d*`);`

on f és la funció i $[a, b] \times [c, d]$ el rectangle on està definida. Per comoditat, podem definir prèviament aquest rectangle, i fer-lo servir en totes les crides a `plot3d`:

```
> U := a..b,c..d;
> plot3d(f,U);
```

Es poden representar diverses funcions alhora, només cal incloure-les entre claus `{...}` dins la crida a `plot3d`; això val en general amb totes les altres comandes gràfiques.

Hi ha una sintaxi alternativa per a `plot3d`, i és convenient conèixer les dues versions, ja que cada una té els seus avantatges. Es tracta de fer servir expressions en comptes de funcions. El mateix cas de l'apartat anterior, però fet amb expressions, aniria així:

`plot3d(`*expr*`,`*x*`=`*a*`..`*b*`,`*y*`=`*c*`..`*d*`);`

on *expr* és l'expressió de la funció en termes de x i y (vegeu l'apèndix). En aquest cas, més que definir el conjunt on dibuixem la gràfica, donem el rang que prenen les variables. Per comoditat, es pot definir prèviament aquest rang, i fer-lo servir en totes les crides a `plot3d`:

```
> rangxy := x=a..b,y=c..d;
> plot3d(f(x,y),rangxy);
```

Un cop tinguem la gràfica dibuixada, podem fer moltes coses per tal de veure-la millor. Si arrosseguem la imatge tridimensional amb el ratolí, podem fer-la girar per a millorar-ne perspectiva. Prement el botó dret del ratolí sobre la figura, apareixen diverses opcions amb què podem experimentar. Algunes d'elles

poden activar-se també amb uns botonets que surten a la barra de menús quan tenim seleccionada la imatge.

Algunes de les opcions més interessants són les següents. L'opció `axes` permet definir l'estil amb què volem els eixos. L'opció `style` defineix la manera de dibuixar la trama de la superfície. L'opció `grid` especifica quants punts volem que tingui aquesta trama; com més punts tinguem millor es veurà la superfície, però la composició de la gràfica serà més lenta i consumirà més memòria. L'opció `view` especifica el rang de valors de la funció que es representaran gràficament. Altres opcions útils són `scaling` i `orientation`. Més que explicar-les totes detalladament, és més interessant descobrir *in situ* el seu funcionament.

Si estem segurs de quines opcions volem, podem especificar-les en la pròpia crida a `plot3d`. Per exemple

```
> plot3d(f,U,axes=framed,scaling=constrained);
```

Si fem això, potser sen's farà pesat haver de teclejar en cada crida a `plot3d` la mateixa llarga cadena d'opcions. Afortunadament, hi ha una funció que ens permet especificar d'una vegada per totes les nostres opcions preferides. Es tracta de `setoptions3d`, que pertany a la llibreria `plots`. Per exemple,

```
> with(plots):
> setoptions3d(axes=framed,scaling=constrained);
```

fa que `axes=framed` i `scaling=constrained` siguin les opcions per defecte. Per a veure una llista enorme amb totes les opcions possibles que podem fer servir, es pot escriure

```
> ?plot3d[option]
```

Finalment esmentarem les comandes

`contourplot(`f`,`$a..b$`,`$c..d$`);`

que dibuixa algunes corbes de nivell de f sobre el domini especificat, i

`implicitplot(`f`=`k`,`$a..b$`,`$c..d$`);`

que dibuixa la corba de nivell k de la funció f dins el rectangle especificat.

Cal observar que l'ús d'aquestes dues comandes també requereix la càrrega prèvia de la llibreria `plots`. Hi ha altres comandes gràfiques en Maple, com ara `plot`, `implicitplot3d` i `spacecurve`, de les quals parlarem més endavant.

1.3 Tutorial

1.3.1 Gràfiques de funcions de dues variables

El primer que farem és dibuixar la gràfica d'una funció, per exemple $f(x,y) = \dfrac{\sin(x^2+y^2)}{1+x^2+y^2}$ sobre el rectangle $U = [-3,3] \times [-3,3]$. Primerament definim la funció i el rectangle, i tot seguit en dibuixem la gràfica:

```
> f:=(x,y)->sin(x^2+y^2)/(1+x^2+y^2);
> U:=-3..3,-3..3;
> plot3d(f,U);
```

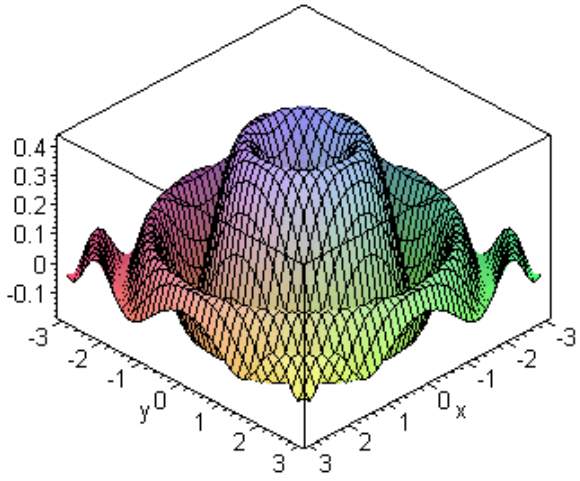

Figura 1.1: Gràfica de f sobre el rectangle $[-3,3]\times[-3,3]$

1.3.2 Corbes de nivell

Anem a estudiar la funció $g(x,y)=(x^2+3y^2)e^{1-x^2-y^2}$. Per exemple, podem dibuixar-ne la gràfica tallada a l'alçada $z=2$ amb l'opció `view`:

```
> g:=(x,y)->(x^2+3*y^2)*exp(1-x^2-y^2);
> U:=-2..2,-2..2;
> plot3d(g,U,view=0..2);
```

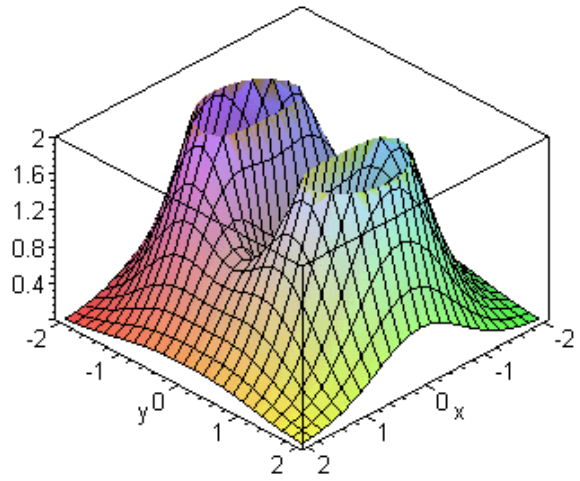

Figura 1.2: Gràfica de g tallada a $z=2$

Si ens sembla que la figura no és prou precisa, podem afinar-ne els detalls amb l'opció `grid`:

```
> plot3d(g,U,view=0..2,grid=[50,50]);
```

Notem que, si girem la gràfica fins a tenir-ne una vista superior (o inferior), llavors veurem la corba de nivell $g(x,y)=2$. De tota manera, el millor procediment per a veure aquesta corba de nivell és entrar

```
> implicitplot(g(x,y)=2,x=-2..2,y=-2..2);
```

Si volem veure diverses corbes de nivell de la funció podem aplicar la comanda `contourplot`:

```
> contourplot(g(x,y),x=-2..2,y=-2..2,scaling=constrained);
```

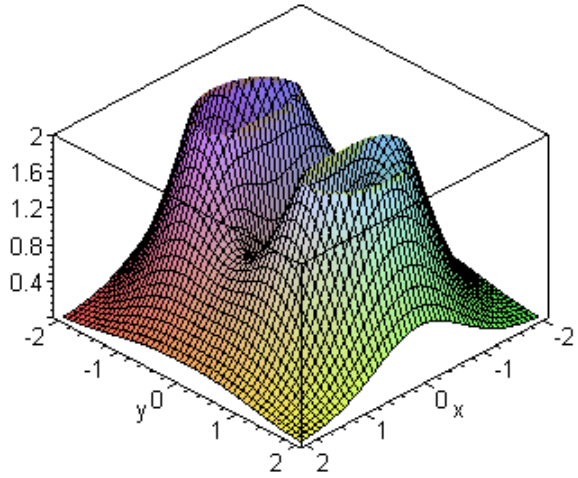

Figura 1.3: La gràfica anterior amb millor `grid`

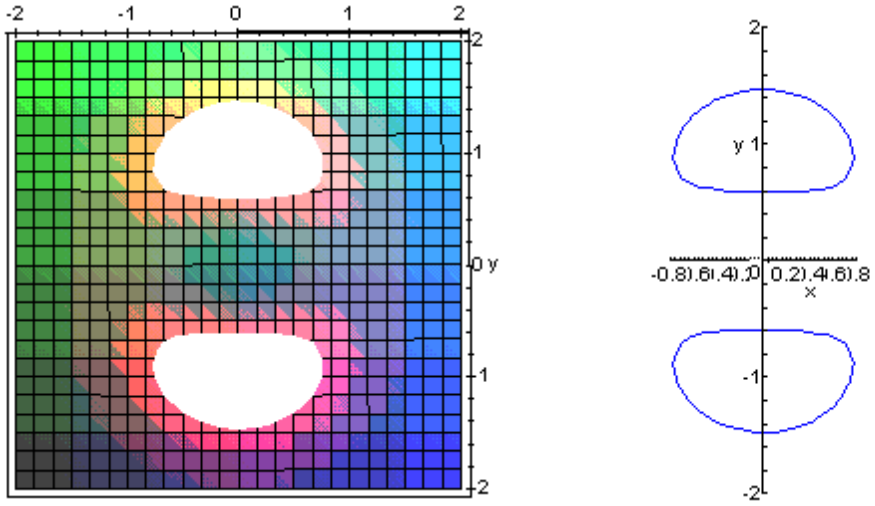

Figura 1.4: Corba de nivell $g(x, y) = 2$, girant la gràfica anterior i usant `implicitplot`

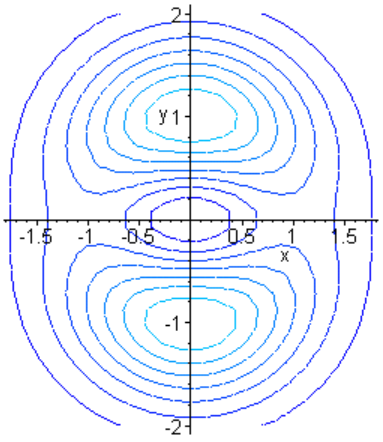

Figura 1.5: Diverses corbes de nivell de g

2 Límits i continuïtat

2.1 Introducció

Les possibilitats d'apropar-se a un punt sobre el pla són molt més variades que sobre la recta real. De fet, hi ha una infinitat de rectes del pla que passen per un punt, i si això no és prou podem acostar-nos a un punt seguint corbes tan variades com vulguem.

Si $f\colon A \to \mathbf{R}$ és una funció, i p_0 és un punt d'acumulació de A, recordem que es diu que l és el *límit* de $f(p)$ quan p tendeix a p_0, i s'escriu $\lim_{p\to p_0} f(p) = l$, quan per a tota $\varepsilon > 0$ existeix $\delta > 0$ tal que, si $d(p, p_0) < \delta$, llavors $d(f(p), l) < \varepsilon$. Si p_0 és del domini de f i l coincideix amb $f(p_0)$, f és *contínua* en p_0.

Aquesta definició de límit no posa restriccions a la forma d'apropar-se al punt p_0. En aquest sentit, podem considerar els anomenats límits direccionals: si $u \neq 0$ és un vector, la recta parametritzada per $p_0 + tu$ passa per p_0, i potser podem calcular el *límit direccional* $\lim_{t\to 0} f(p_0 + tu)$. Per exemple, si f és una funció de dues variables ens podem apropar a un punt amb una direcció de pendent m, o sigui, segons el vector director $(1, m)$.

Més generalment, ens podem apropar a un punt seguint corbes qualssevol que passin per aquest punt.

Recordem alguns resultats bàsics. Si existeix el límit de f en un punt, tots els límits direccionals en aquest punt també existeixen i valen el mateix. Tanmateix, l'existència i igualtat de tots els límits direccionals no implica l'existència del límit. En aquest cas, per a demostrar la no existència del límit es pot recórrer a calcular el límit de la funció al llarg de corbes apropiades.

2.2 Comandes Maple

En Maple tenim una funció que calcula límits unidimensionals. La comanda

`limit(`*expr*`,`*x*`=`*a*`);`

on *expr* és una expressió en la variable x, calcula el límit de *expr* quan x tendeix a a.

Recordem com es dibuixaven gràfiques de funcions d'una variable:

`plot(`*expr*`,`*x*`=`*a*`..`*b*`);`

si *expr* és una expressió en la variable x, dibuixa la gràfica de la funció corresponent, sobre l'interval $[a, b]$.

Una eina gràfica que utilitzarem per a estudiar límits és

`animate(`*expr*`,`*x*`=`*a*`..`*b*`,`*t*`=`*t0*`..`*tf*`);`

on *expr* és una expressió en les variables x i t. Aquesta comanda crea una successió de plots corresponents a diversos valors del paràmetre t. Per a veure la succesió de tots els plots s'ha de prémer un botó "play" ▷ que apareix. Si volem que l'animació es repeteixi indefinidament, hi ha un botó "repeat" ↻. L'opció `frames` permet especificar el nombre d'imatges de l'animació, que és per defecte 16. Per a usar `animate` cal carregar la llibreria `plots`.

2.3 Tutorial

2.3.1 Límits direccionals

Anem a estudiar la funció $f(x, y) = \dfrac{2xy}{x^2 + y^2}$ al voltant de l'origen. Si en dibuixem la gràfica de seguida veiem que té problemes a l'origen:

```
> f:=(x,y)->(2*x*y)/(x^2+y^2);
> plot3d(f(x,y),x=-2..2,y=-2..2);
```

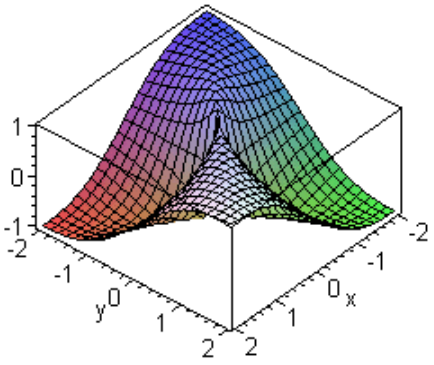

Figura 2.1: Gràfica de f al voltant de l'origen

Podem dibuixar la restricció de f a diverses rectes passant per l'origen, per exemple les funcions $f(x,x)$ i $f(x,-x)$:

```
> plot(f(x,x),x=-2..2);
> plot(f(x,-x),x=-2..2);
```

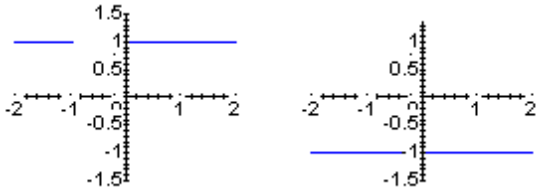

Figura 2.2: Gràfiques de $f(x,x)$ i de $f(x,-x)$

En les gràfiques resultants es veu clarament que el límit de f quans ens acostem a l'origen seguint la recta $y=x$ és 1, i el límit seguint la recta $y=-x$ és -1. Tot i que en aquest cas no cal, també podríem utilitzar la funció de càlcul de límits del Maple,

```
> limit(f(x,x),x=0); limit(f(x,-x),x=0);
```

que ens dóna els mateixos límits.

Atès que obtenim diferents límits direccionals, concloem que la funció no té límit a l'origen. Aquest fet queda també evidenciat si dibuixem les corbes de nivell de f al voltant de l'origen (figura 2.3):

```
> with(plots):
> contourplot(f(x,y),x=-2..2,y=-2..2,scaling=constrained);
```

2.3.2 Límits al llarg de corbes

Considerem ara la funció $g(x,y)=\dfrac{x^2y}{x^4+y^2}$. Si en dibuixem la gràfica al voltant de l'origen observem que no és gaire clara l'existència de límit.

```
> g:=(x,y)->x^2*y/(x^4+y^2);
> plot3d(g(x,y),x=-1..1,y=-1..1,style=patchcontour);
```

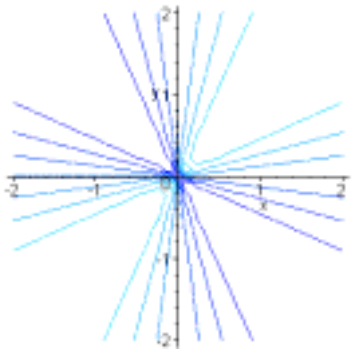

Figura 2.3: Corbes de nivell de f al voltant de l'origen

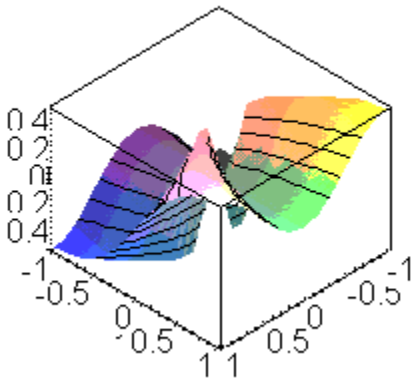

Figura 2.4: Gràfica de g al voltant de l'origen

Podem restringir la funció a diferents rectes passant per l'origen; per exemple, les rectes de vector director $(\cos t, \sin t)$, per a diferents valors de t. Una manera ràpida de fer-ho és amb `animate`:

```
> with(plots):
> animate(g(r*cos(t),r*sin(t)),r=-1..1,t=0..Pi);
```

(Atenció: l'ordre amb què especifiquem els rangs és important; en aquest cas, r és la variable independent de cada imatge de l'animació, mentre que t és la variable que parametritza les diferents imatges.)

S'observa que sobre totes aquestes rectes el límit existeix i és el mateix, 0. Tanmateix, si calculem la funció al llarg de les paràboles $y = \lambda x^2$ observarem límits diferents quan $x \to 0$ (figura 2.5):

```
> animate(g(x,lambda*x^2),x=-0.5..0.5,lambda=0..5);
```

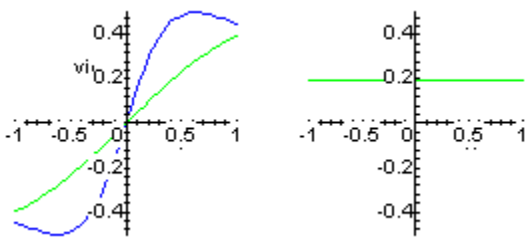

Figura 2.5: Gràfica de g sobre rectes i sobre paràboles

3 Diferenciabilitat

3.1 Introducció

En el càlcul d'una variable, la interpretació geomètrica de l'existència de la derivada d'una funció és l'existència d'una recta tangent al graf de la funció. És aquesta la idea que permet estendre el concepte de diferenciabilitat a funcions de diverses variables.

Donada una aplicació $y = f(x)$, amb $f\colon \mathbf{R}^m \to \mathbf{R}^n$, i un punt x_0 del domini de f, es diu que f és *diferenciable* en x_0 si existeix una aplicació lineal $T\colon \mathbf{R}^m \to \mathbf{R}^n$ tal que

$$\lim_{x\to x_0} \frac{f(x) - f(x_0) - T\cdot(x-x_0)}{\|x-x_0\|} = 0.$$

Aquesta aplicació lineal T es diu *derivada* o *diferencial* de f en x_0, i es representa per $\mathrm{D}f(x_0)$ o per $f'(x_0)$.

La diferenciabilitat també es pot expressar així: $f(x) = f(x_0) + T\cdot(x-x_0) + o(\|x-x_0\|)$, és a dir, $f(x_0) + T\cdot(x-x_0)$ és una "aproximació lineal" de la funció f al voltant de x_0. En el cas d'una funció escalar de dues variables, podrem visualitzar de manera efectiva la tangència entre el graf de la funció i la seva aproximació lineal.

La relació entre el concepte de derivada en termes d'aproximació lineal i la idea més familiar de derivada com a límit d'un quocient ve donada pel concepte de derivada direccional. Si $\mathbf{u}$ és un vector de $\mathbf{R}^m$, el límit

$$\lim_{t\to 0} \frac{f(x_0 + t\mathbf{u}) - f(x_0)}{t},$$

si existeix, s'anomena *derivada direccional* de f en x_0 segons el vector $\mathbf{u}$, i es representa per $\mathrm{D}_{\mathbf{u}}f(x_0)$ o per $f'(x_0;\mathbf{u})$; a vegades interessa que $\mathbf{u}$ sigui unitari, però en general això no és necessari. Si f és diferenciable en x_0, totes les derivades direccionals hi existeixen, i es poden calcular per $\mathrm{D}_{\mathbf{u}}f(x_0) = \mathrm{D}f(x_0)\cdot\mathbf{u}$.

Un cas particular s'obté prenent per $\mathbf{u}$ els vectors $\mathbf{e}_i$ de la base canònica de $\mathbf{R}^m$: són les *derivades parcials* de f en x_0, $\mathrm{D}_i f(x_0)$, també escrites $\partial f(x_0)/\partial x^i$.

Recordem també que la diferenciabilitat, etc., de la funció f ve donada per la diferenciabilitat de les seves funcions components f^j.

Si en un punt existeixen totes les derivades parcials $\mathrm{D}_i f^j(x_0)$, aquestes formen la *matriu jacobiana* de f en x_0,

$$\mathrm{J}f(x_0) = \begin{pmatrix} \partial f^1(x_0)/\partial x^1 & \cdots & \partial f^1(x_0)/\partial x^m \\ \vdots & \ddots & \vdots \\ \partial f^n(x_0)/\partial x^1 & \cdots & \partial f^n(x_0)/\partial x^m \end{pmatrix}.$$

Si la funció és diferenciable en x_0, la matriu jacobiana existeix i és de fet la matriu de l'aplicació lineal $\mathrm{D}f(x_0)$ en les bases canòniques. Tanmateix, l'existència de les derivades parcials o direccionals no assegura la diferenciabilitat; ara bé, si totes les derivades parcials són contínues (f es diu llavors de classe C^1) f és diferenciable. Recordem, però, que hi ha funcions diferenciables amb derivada no contínua.

3.2 Comandes Maple

Hi ha dues comandes en Maple per derivar. Poden calcular-se derivades parcials de funcions, i derivades d'una expressió respecte una variable.

`D[i](f)`

Aquesta comanda retorna la derivada parcial de la funció f respecte la variable i-èsima. És a dir, retorna una funció, del mateix nombre de variables que f.

Si preferim treballar amb expressions, que sovint és més còmode, llavors tenim la comanda `diff`:

`diff(`*expr*`,`*x*`)`

que calcula la derivada parcial de l'expressió *expr* respecte la variable x. Si es posen més variables, es calculen derivades d'ordre superior.

Fent ús només d'aquestes comandes, ja podem calcular els operadors diferencials típics, com el gradient o la hessiana. Però la llibreria `linalg` ja té aquestes funcions incorporades. Per exemple,

`grad(`*expr*`,`*vars*`);`

calcula el gradient –la diferencial considerada com a vector– de la funció escalar representada per l'expressió *expr*, considerada com a funció de les variables de la llista *vars*. La matriu hessiana s'obté de manera semblant amb

`hessian(`*expr*`,`*vars*`);`

Per a una funció vectorial, representada per una llista d'expressions *exprs*, podem calcular anàlogament la matriu jacobiana:

`jacobian(`*exprs*`,`*vars*`);`

Observem que aquestes tres comandes també donen expressions. Si volem funcions, les hem de definir nosaltres mateixos amb l'operador `D`, o bé fent substitucions, tal i com fem en un dels exemples.

3.3 Tutorial

3.3.1 Diferenciabilitat

Estudiarem, com a exemple, la diferenciabilitat en l'origen de la funció

$$f(x,y) = \begin{cases} x^2y^2 \sin\left(\frac{1}{xy}\right) & \text{si } xy \neq 0, \\ 0 & \text{si } xy = 0. \end{cases}$$

Primerament en dibuixem la gràfica en un veïnat del $(0,0)$:

```
> f:=(x,y)->x^2*y^2*sin(1/(x*y));
> plot3d(f(x,y),x=-0.4..0.4,y=-0.4..0.4,grid=[30,30]);
```

Podem fer una ampliació de la zona que ens interessa:

```
> plot3d(f(x,y),x=0..0.5,y=0..0.5,grid=[30,30]);
```

L'aspecte de la gràfica suggereix que la funció és diferenciable en $(0,0)$ i amb diferencial nul·la. Per a comprovar-ho, estudiem l'existència de les derivades parcials en aquest punt. La derivada parcial respecte a x és

$$\mathrm{D}_1 f(0,0) = \lim_{t\to 0} \frac{f(t,0) - f(0,0)}{t} = 0,$$

i anàlogament la parcial respecte a y. Així doncs, si existeix, la diferencial ha de ser nul·la, com suposàvem. Anem a provar que la funció és diferenciable en l'origen.

Prèviament observarem que les derivades parcials sobre els eixos valen 0. Pel que fa a fora dels eixos, calculem les expressions analítiques de les derivades parcials i mirem si són contínues a l'origen:

```
> diff(f(x,y),x); diff(f(x,y),y);
```

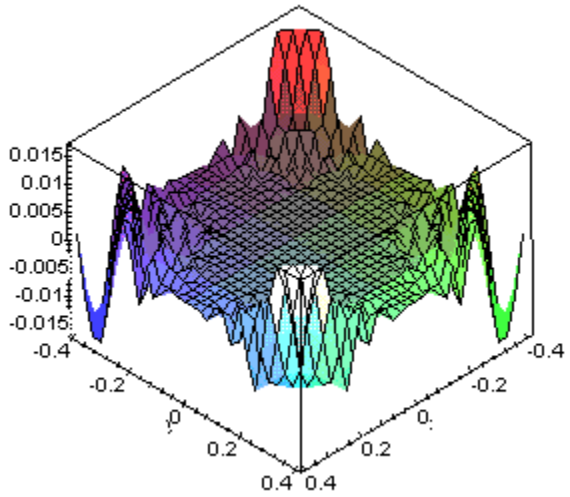

Figura 3.1: Gràfica de $f(x,y)$ al voltant de $(0,0)$

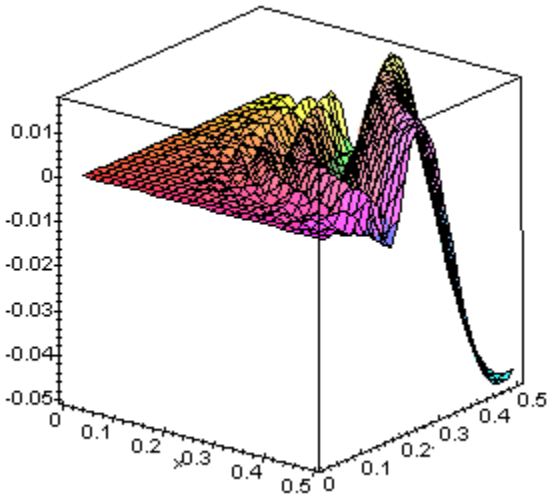

Figura 3.2: Ampliació de la gràfica de $f(x,y)$ al voltant de $(0,0)$

d'on resulta $\dfrac{\partial f}{\partial x} = 2xy^2 \sin\left(\dfrac{1}{xy}\right) - y\cos\left(\dfrac{1}{xy}\right)$, $\dfrac{\partial f}{\partial y} = 2yx^2 \sin\left(\dfrac{1}{xy}\right) - x\cos\left(\dfrac{1}{xy}\right)$. Si fem tendir $(x,y) \to (0,0)$, és clar que tendeixen a zero les dues derivades parcials (cada terme és el producte d'una funció amb límit zero per una funció fitada), per tant són contínues a l'origen i, com a conseqüència, la funció f hi és diferenciable.

3.3.2 Aproximació lineal en un punt

Considerem ara la funció $g(x,y) = x^2 + y^2$. Podem calcular-ne la diferencial, i amb ella l'aproximació lineal de g en el punt $(1/2, 1/2)$:

```
> g:=(x,y)->x^2+y^2;
> Dg:=(x,y)->subs({X=x,Y=y},grad(g(X,Y),[X,Y]));
> x0:=1/2; y0:=1/2;
> lg:=(x,y)->g(x0,y0)+Dg(x0,y0)[1]*(x-x0)+Dg(x0,y0)[2]*(y-y0);
```

Observem que aquí hem volgut definir la diferencial Dg com a funció del punt (x, y): això no es pot fer directament amb la comanda `grad`, sinó que cal aplicar-la a l'expressió `g(X,Y)`, on `X` i `Y` són unes variables auxiliars; posteriorment, mitjançant la comanda `subs`, hem substituït les variables auxiliars per les variables "autèntiques", `x` i `y`, que usem per a definir la funció `Dg`.

Si representem simultàniament les gràfiques de g i la seva aproximació lineal, veurem que són tangents:

```
> plot3d({g(x,y),lg(x,y)},x=0..1,y=0..1);
```

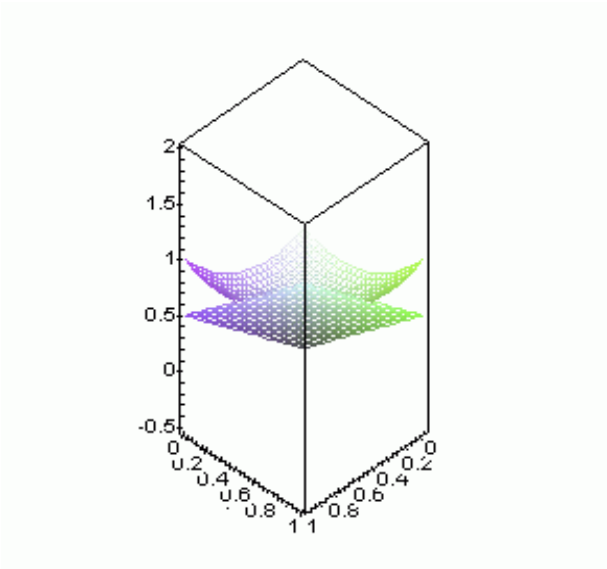

Figura 3.3: Gràfica de g i la seva aproximació lineal al voltant de $(1/2, 1/2)$

4 Corbes i superfícies

4.1 Introducció

Una *superfície regular* de $\mathbf{R}^3$ és un subconjunt $S \subset \mathbf{R}^3$ que és localment com un pla: al voltant de cada punt $p \in S$, hi ha un canvi de coordenades de $\mathbf{R}^3$ que transforma la superfície en un fragment de pla. Semblantment, una *corba regular* de $\mathbf{R}^3$ és un subconjunt $C \subset \mathbf{R}^3$ que és localment com una recta.

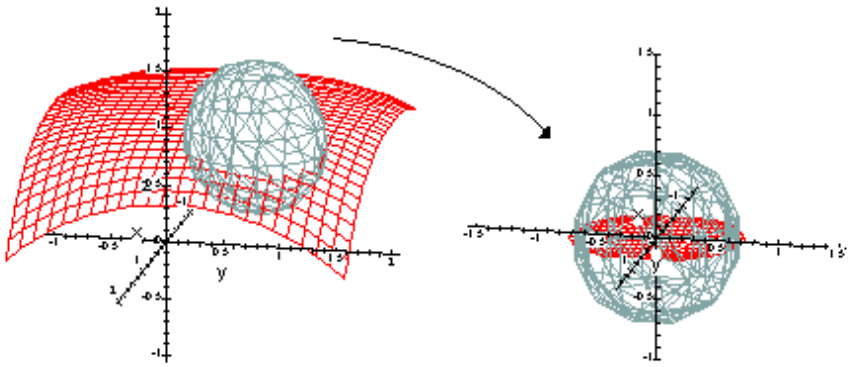

Figura 4.1: Una superfície i un aplanament local d'ella al voltant d'un punt

De la mateixa manera que un pla de $\mathbf{R}^3$ es pot descriure de forma implícita o paramètrica, també es pot fer amb una superfície, per bé que en general aquesta descripció serà només local.

Descripció implícita Per exemple, si $F\colon W \to \mathbf{R}$ és una funció de classe C^1 definida en un obert $W \subset \mathbf{R}^3$, sigui

$$S = \{x \in W \mid F(x) = 0\} = F^{-1}(0),$$

suposant que $S \neq \emptyset$; si en tot $x \in S$ el rang de la diferencial de F és el màxim possible (en aquest cas això significa que rang $\mathrm{D}F(x) = 1$), llavors S és una superfície regular. Es diu que S està *descrita implícitament* per l'equació $F(x) = 0$.

Un enunciat similar s'obté prenent una equació amb dues variables, o un sistema de dues equacions amb tres variables; llavors s'obté una corba en $\mathbf{R}^2$ o en $\mathbf{R}^3$, respectivament.

Cas particular: la gràfica d'una funció Si $f\colon U \to \mathbf{R}$ és de classe C^1 sobre un obert $U \subset \mathbf{R}^2$, la seva gràfica

$$S = \{(x, y, z) \mid (x, y) \in U, z = f(x, y)\}$$

és una superfície dins $\mathbf{R}^3$; es diu a vegades que S ve *descrita explícitament* per la funció $z = f(x, y)$. Observem que, de fet, S es pot descriure implícitament per l'equació $F(x, y, z) := z - f(x, y) = 0$.

De manera semblant es poden construir corbes en el pla o en l'espai.

Descripció paramètrica Si $g\colon U \to \mathbf{R}^3$ és de classe C^1 en un obert $U \subset \mathbf{R}^2$, i en un punt $u_0 \in U$ el rang de la diferencial és màxim (en aquest cas això vol dir que rang $\mathrm{D}g(u_0) = 2$), llavors existeix un obert $U' \subset U$ contenint u_0 tal que

$$S = g(U') \subset \mathbf{R}^3$$

és una superfície regular. Es diu que g és una parametrització de S, i que S està *descrita paramètricament* per g.

De manera semblant, si I és un obert de $\mathbf{R}$, una aplicació $g\colon I \to \mathbf{R}^3$ s'anomena *corba parametritzada*. En condicions apropiades, la imatge de g serà una corba dins $\mathbf{R}^3$.

Observació important: aquest és un resultat local, en general la imatge sencera $g(U)$ pot no ser una superfície o una corba regular, per més que g pugui tenir bones propietats (després en veurem exemples).

4.2 Comandes Maple

Anem a veure com es dibuixen tots els casos anteriors de corbes i superfícies en Maple. Algunes de les comandes ja les coneixem, però les incloem igualment per completesa. La majoria de les comandes necessiten la llibreria `plots`.

4.2.1 Corbes en el pla

Recordem que hi ha tres maneres de definir una corba plana. Cada una d'elles té la comanda Maple corresponent. La seva sintaxi és força clara. Usualment s'entra l'expressió de la funció que defineix la corba, i es diuen els rangs que han de prendre les variables.

Gràfica d'una funció $\mathbf{R} \to \mathbf{R}$:

`plot(`*expr*`,`*x*`=`*a*`..`*b*`);`

Corba parametritzada (imatge d'una funció $\mathbf{R} \to \mathbf{R}^2$):

`plot([`*expr1*`,`*expr2*`,`*t*`=`*t0*`..`*tf*`]);`

Corba definida de forma implícita per una equació de dues variables (corba de nivell d'una funció $\mathbf{R}^2 \to \mathbf{R}$):

`implicitplot(`*equació*`,`*x*`=`*a*`..`*b*`,`*y*`=`*c*`..`*d*`);`

Les dues primeres comandes tenen la seva versió amb funcions en comptes d'expressions:

`plot(`*f*`,`*a*`..`*b*`);` `plot([`*f*`,`*g*`,`*t0*`..`*tf*`]);`

4.2.2 Superfícies en l'espai

També hi ha tres maneres de definir una superfície en l'espai. El funcionament d'aquestes comandes és semblant al de les corbes planes, però amb una variable més. Val a dir que la comanda `implicitplot3d` no fa dibuixos gaire bonics, i és molt lenta. Aconsellem evitar-la sempre que sigui possible (per exemple, parametritzant la superfície que volem representar).

Gràfica d'una funció $\mathbf{R}^2 \to \mathbf{R}$:

`plot3d(`*expr*`,`*x*`=`*a*`..`*b*`,`*y*`=`*c*`..`*d*`);`

Superfície parametritzada (imatge d'una funció $\mathbf{R}^2 \to \mathbf{R}^3$):

`plot3d([`*expr1*`,`*expr2*`,`*expr3*`],`*u*`=`*a*`..`*b*`,`*v*`=`*c*`..`*d*`);`

Superfície definida de forma implícita per una equació de tres variables (superfície de nivell d'una funció $\mathbf{R}^3 \to \mathbf{R}$):

`implicitplot3d(`*equació*`,`*x*`=`*a*`..`*b*`,`*y*`=`*c*`..`*d*`,`*z*`=`*e*`..`*f*`);`

Igual que en les corbes, les dues primeres comandes tenen la seva versió funcional:

`plot3d(`*f*`,`*a*`..`*b*`,`*c*`..`*d*`);` `plot3d([`*f*`,`*g*`,`*h*`],`*u0*`..`*uf*`,`*v0*`..`*vf*`);`

4.2.3 Corbes en l'espai

Podem descriure una corba en l'espai de manera paramètrica, o bé com a intersecció de dues superfícies. Només farem servir la descripció paramètrica (com a imatge d'una funció $\mathbf{R} \to \mathbf{R}^3$):

```
spacecurve([f(t),g(t),h(t)],t=t0..tf);
```

Una comanda amb la mateixa sintaxi, però que de vegades fa dibuixos més bonics, és `tubeplot`.

4.3 Tutorial

4.3.1 Corbes al pla

L'equació $x^{2/3} + y^{2/3} = 1$ defineix una astroide. Aquesta corba es pot parametritzar per $\gamma(t) = (\cos^3 t, \sin^3 t)$, $0 \leq t < 2\pi$. Per tant, es podrà representar gràficament (figura 4.2) per les dues comandes següents:

```
> plot([cos(t)^3,sin(t)^3,t=0..2*Pi]);
> implicitplot(surd(x,3)^2+surd(y,3)^2 = 1,x=-1..1,y=-1..1);
```

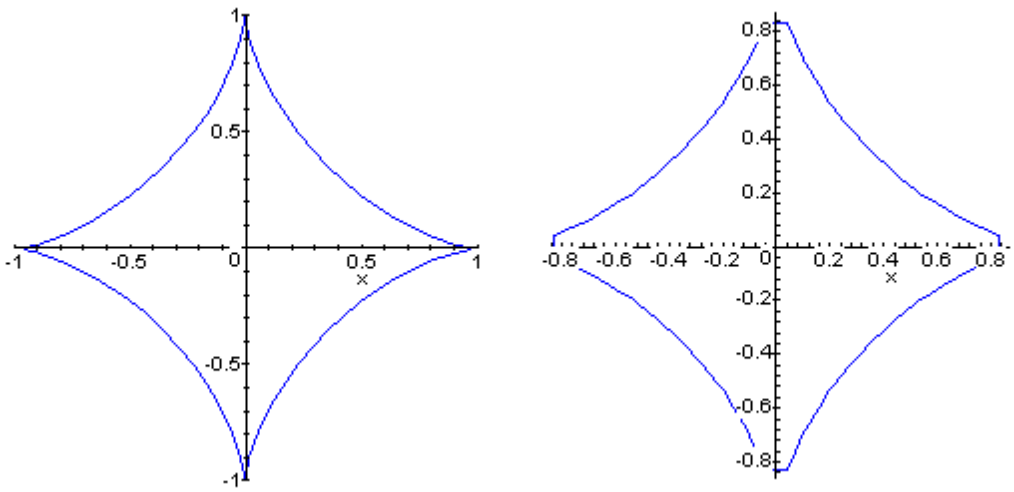

Figura 4.2: Astroide dibuixada a partir de la parametrització i de la forma implícita

El resultat de la segona és més pobre; per tal de millorar-lo es pot canviar, si es vol, el `grid`.

Observem que l'astroide té quatre punxes, i doncs no és una corba regular en aquests punts. No és casual que són aquests els punts on la parametrització γ té derivada nul·la, i també els punts de la corba on la funció $F(x, y) = x^{2/3} + y^{2/3}$ no és diferenciable.

4.3.2 Corbes a l'espai

La corba parametritzada $\gamma(t) = (\cos t, \sin t, t)$ descriu una hèlix. Podem veure'n un fragment (figura 4.3):

```
> spacecurve([cos(t),sin(t),t],t=0..6*Pi,numpoints=100);
```

Observem que l'hèlix també es pot descriure implícitament amb el sistema $x^2 + y^2 = 1$, $x = \cos z$. Si ho volem, podem representar gràficament les superfícies definides per aquestes equacions; la seva intersecció és l'hèlix.

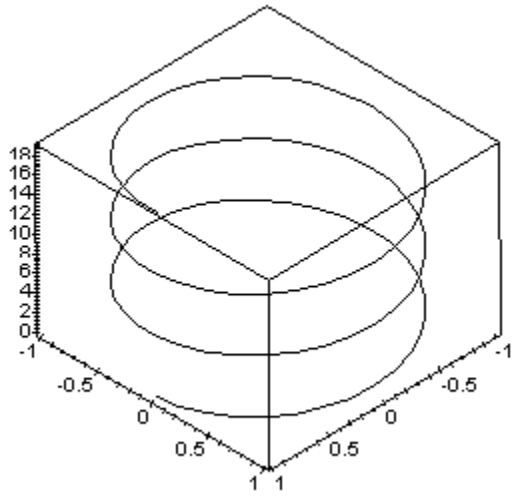

Figura 4.3: Un fragment d'hèlix

4.3.3 Superfícies a l'espai

L'equació $x^2 + y^2 + z^2 = 1$ defineix una esfera. Per tant, es pot representar gràficament per

```
> implicitplot3d(x^2+y^2+z^2=1,x=-1..1,y=-1..1,z=-1..1
  style=patchcontour,scaling=constrained,axes=boxed);
```

Una representació paramètrica de l'esfera és la donada pels angles de les coordenades esfèriques, colatitud i longitud:

```
> g:=(u,v)->[cos(v)*sin(u),sin(v)*sin(u),cos(u)];
> plot3d(g(u,v),u=0..Pi,v=0..2*Pi,scaling=constrained,axes=boxed);
```

Els gràfics resultants són a la figura 4.4.

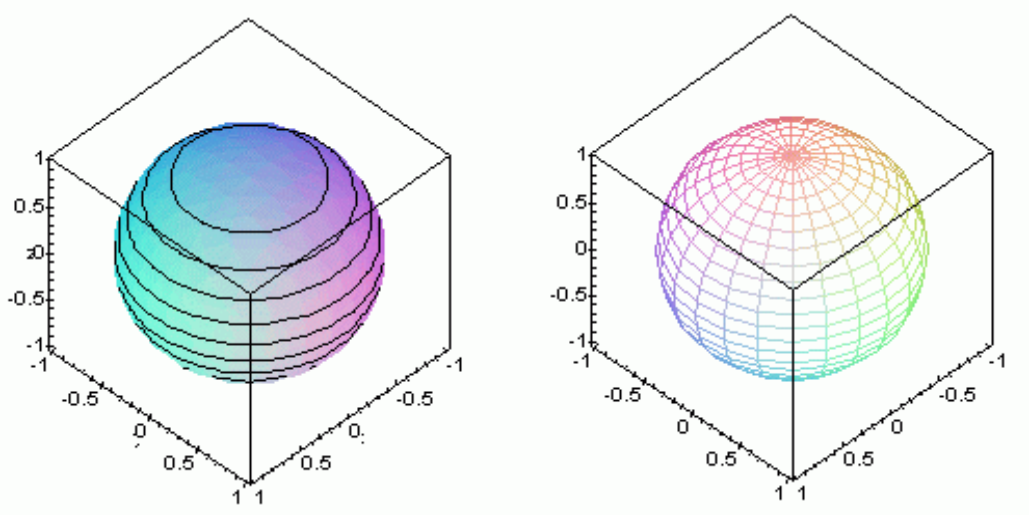

Figura 4.4: Esfera dibuixada amb `implicitplot3d` i amb la parametrització `g` donada pels angles; observeu, en aquest cas, que les línies coordenades corresponen als meridians i als paral·lels

4.3.4 Més exemples

Corba en forma de 8 Considerem la corba parametritzada al pla $c(t) = (\sin 2t, \sin t)$, $0 < t < 2\pi$. La parametrització és regular i injectiva, i tanmateix la imatge $C = c(]0, 2\pi[)$ no és una corba regular. La raó d'això és que, quan t tendeix als extrems de l'interval, $c(t)$ tendeix cap a $(0, 0)$. El dibuix (figura 4.5) ens ho aclareix de seguida:

```
> c := t -> sin(2*t),sin(t);
> plot([c(t),t=0..2*Pi]);
```

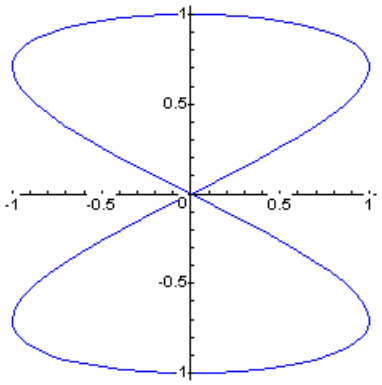

Figura 4.5: La imatge de c és un 8

Corba densa en un tor Considerem la parametrització $g(u, v) = ((3+\cos v)\cos u, (3+\cos v)\sin u, \sin v)$; la seva imatge és un tor a l'espai, de radi gran 3 i radi petit 1. Observem que, com que les funcions són de període 2π, podem suposar que u, v estan dins $[0, 2\pi[$; de fet, aquestes variables són angles que determinen un punt del tor.

Podem dibuixar corbes damunt aquest tor de manera molt simple: un camí en l'espai dels paràmetres (u, v), transportat per g, dóna un camí en el tor. Per exemple, la recta $v = mu$, de pendent m, ens donarà una corba:

```
> g:=(u,v)->[(3+cos(v))*cos(u),(3+cos(v))*sin(u),sin(v)];
> pl_tor:=plot3d(g(u,v),u=0...2*Pi,v=0...2*Pi,style=wireframe):
> m:=5;
> pl_cor:=spacecurve(g(u,m*u),u=0..2*Pi,color=black,numpoints=200):
> display({pl_tor,pl_cor},scaling=constrained);
```

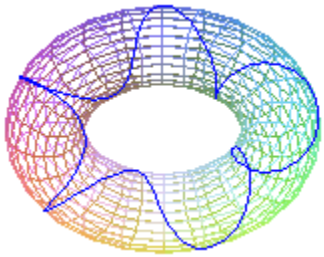

Figura 4.6: Una corba tancada damunt un tor ($m = 5$)

Aquí hem pres $m = 5$, així doncs, sobre la corba, quan la coordenada u fa una volta completa, la coordenada v en fa 5, i la corba és tancada (figura 4.6). Per tant, si en lloc de prendre $u \in [0, 2\pi]$ prenguéssim $u \in \mathbf{R}$ (cosa que el Maple no farà, és clar), la corba seria la mateixa.

I si prenguéssim $m = 4/3$? La corba es tancaria al cap de 3 voltes de u; encara que fem 50 voltes, tindrem una corba tancada (figura 4.7):

```
> m:=4/3;
> pl_cor:=spacecurve(g(u,m*u),u=0..100*Pi,color=blue,numpoints=1000):
```

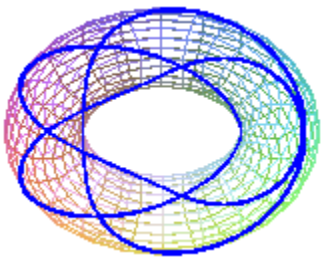

Figura 4.7: Una altra corba tancada damunt un tor ($m = 4/3$)

És clar que aquest argument serveix per a qualsevol valor racional de m.

Provem ara $m = \sqrt{2}$:

```
> m2:=sqrt(2);
> pl_cor:=spacecurve(g(u,m2*u),u=0..100*Pi,color=blue,numpoints=1000):
```

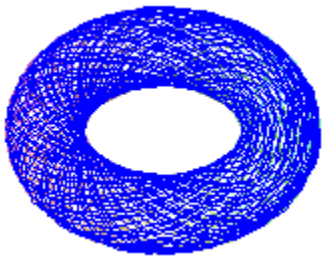

Figura 4.8: Una altra corba, però no tancada ($m = \sqrt{2}$)

El dibuix (figura 4.8) ens fa sospitar què passa. De fet, es pot demostrar que si m és irracional, malgrat que la parametrització $u \mapsto g(u, mu)$, $u \in \mathbf{R}$, és C^∞, injectiva i regular, la imatge no és una corba regular, sinó una "corba" densa en el tor.

5 Polinomis de Taylor

5.1 Introducció

A l'hora de calcular el valor numèric d'una funció en un punt, la màxima facilitat es dóna quan la funció és polinòmica. L'existència de calculadores no ens ha de fer oblidar que, fet i fet, aquestes fan sumes i productes, i les altres funcions que ofereixen estan al capdavall implementades essencialment amb sumes i productes, és a dir, amb polinomis.

Interessa, doncs, obtenir aproximacions polinòmiques de les funcions. Aquestes aproximacions poden obtenir-se a partir de diferents criteris, en funció de l'aplicació que se n'hagi de fer. Una manera d'aproximar funcions és amb el polinomi de Taylor.

Considerem una funció de n variables $f: U \to \mathbf{R}$, un enter $k \geq 1$, i un punt $x_0 \in U$. Se suposa que f és almenys de classe C^k. Llavors existeix un únic polinomi P de grau $\leq k$ tal que les derivades parcials d'ordre $\leq k$ de f i P en el punt x_0 coincideixen. Si escrivim $f = P + R$, la "proximitat" entre f i el polinomi de Taylor P s'expressa en termes de la "petitesa" del terme complementari R; aquest compleix que $R(x) = o(\|x - x_0\|^k)$. Quan f és de classe C^{k+1} es pot expressar el terme complementari de diverses maneres que ara no recordarem.

És clar que com més gran sigui el grau del polinomi, millor aproximació obtindrem, però això té el seu cost de càlcul. Remarquem finalment que aquesta aproximació és local: a una certa distància del punt x_0 la diferència entre el valor de f i el de P pot ser molt elevada. Si es vol aproximar de manera uniforme en tota una regió cal recórrer a altres mètodes d'aproximació.

5.2 Comandes Maple

`mtaylor(`*expr*`,`*vars*`,`*n*`);`

La comanda `mtaylor` calcula el polinomi de Taylor d'una funció a l'origen de coordenades. Aquí *expr* és l'expressió de la funció, *vars* és una llista de variables, i n és l'ordre del desenvolupament (per exemple, si volem el polinomi de grau ≤ 6, n hauria de ser `7`). Si volem fer el desenvolupament en un punt diferent de zero, igualem les variables de la llista a les coordenades d'aquell punt.

Notem que per a usar la comanda `mtaylor` cal definir-la prèviament mitjançant

```
> readlib(mtaylor);
```

5.3 Tutorial

Podem utilitzar el Maple per a avaluar gràficament la proximitat entre una funció i el seu polinomi de Taylor.

Considerem, per exemple, la funció $f(x, y) = \sin(xy)$, i calculem el seu polinomi de Taylor de grau ≤ 6 al voltant de l'origen; l'anomenarem P. Transformarem l'expressió de P en una funció, mitjançant `unapply`. Així, entrarem

```
> readlib(mtaylor):
> f := (x,y)->sin(x*y);
> P := unapply(mtaylor(f(x,y),[x,y],7),x,y);
> P(x,y);
```

que dóna com a resultat $xy - \dfrac{1}{6}x^3y^3$.

Comprovem ara la semblança de les dues funcions al voltant de l'origen. A tal fi, podem usar l'opció `color`, de manera que veurem les dues gràfiques amb colors diferents:

```
> rang := x=-1..1,y=-1..1
> pl_f := plot3d(f(x,y),rang, color=blue,style=wireframe):
> pl_P := plot3d(P(x,y),rang, color=red,style=wireframe):
> display({pl_f,pl_P},axes=framed,orientation=[15,75]);
```

La diferència és molt petita. Prenent com a domini de la gràfica rectangles més grans podrem observar més discrepància a mesura que ens allunyem de l'origen. Per exemple, posant com a intervals `-2..2`.

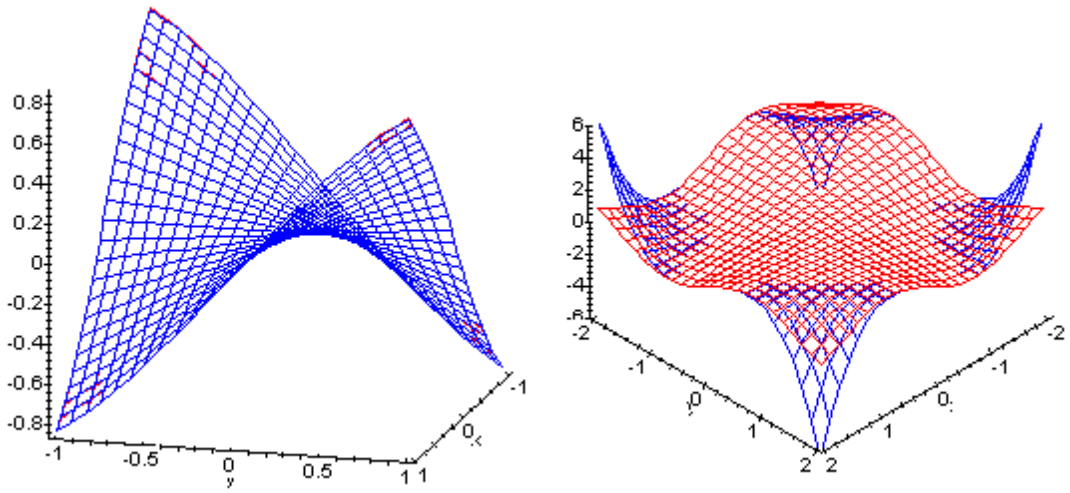

Figura 5.1: Gràfiques de f i P sobre $[-1,1]\times[-1,1]$ i sobre $[-2,2]\times[-2,2]$

Si volem treballar al voltant de, per exemple, $(1,1)$, basta definir

```
> PP := (x,y)->unapply(mtaylor(f(x,y),[x=1,y=1],7),x,y);
```

El resultat és que canvia la zona on l'aproximació és millor. També podem treballar amb un polinomi de Taylor de grau més elevat, per exemple de grau 10. Els resultats gràfics es mostren en la figura 5.2.

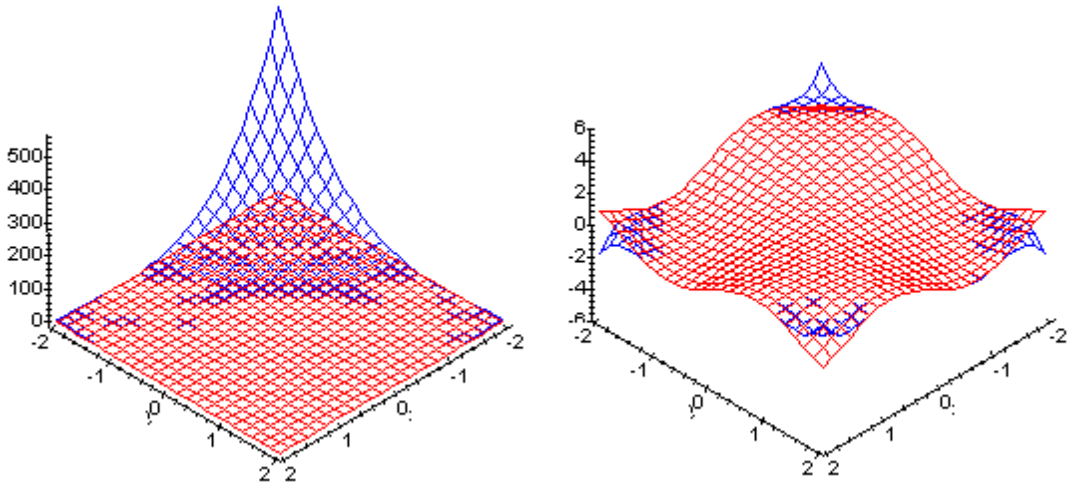

Figura 5.2: Gràfiques de f i el polinomi de Taylor: de grau 6 en $(1,1)$ i de grau 10 en $(0,0)$

6 Extrems locals de funcions

6.1 Introducció

Un dels problemes més freqüents en les aplicacions de les matemàtiques és el càlcul de màxims i mínims de funcions.

Si $f\colon A \to \mathbf{R}$ és una funció, es diu que assoleix un *màxim local* en $x_0 \in A$ si $f(x_0) \geq f(x)$ per als punts x en un veïnat de x_0 dins A. El màxim es diu *estricte* si $f(x_0) > f(x)$ (per a $x \neq x_0$). De manera anàloga es defineix *mínim local*. Un *extrem* és un màxim o mínim.

Suposem que x_0 és un punt interior de A on f és diferenciable; llavors, si f hi té un extrem local, necessàriament x_0 és un *punt crític* de f, és a dir, $\mathrm{D}f(x_0) = 0$. El recíproc és fals: un punt crític pot no ser extrem local; en aquest cas, se'n diu *punt de sella*.

Quan f és de classe C^2, l'estudi de la *matriu hessiana* de f en un punt crític x_0,

$$\mathrm{H}f(x_0) = (\mathrm{D}_i\mathrm{D}_j f(x_0))$$

pot aportar informació que permeti classificar el punt. Més concretament, si $\mathrm{H}f(x_0)$ és definida positiva, x_0 és un mínim local estricte, si $\mathrm{H}f(x_0)$ és definida negativa, x_0 és un màxim local estricte, i si $\mathrm{H}f(x_0)$ és indefinida, x_0 és un punt de sella. En altres casos, és a dir, quan la hessiana és semidefinida, aquest teorema no permet decidir sobre el caràcter del punt crític.

El problema s'ha transformat, doncs, en un problema d'àlgebra lineal: decidir el caràcter d'una matriu simètrica $A = (a_{ij})$. A tal fi es pot usar el *criteri de Sylvester*: es calculen els menors diagonals principals

$$\Delta_1 = a_{11}, \quad \Delta_2 = \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix}, \quad \ldots, \quad \Delta_n = \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{vmatrix}.$$

La matriu és definida positiva sii $\Delta_i > 0$ $(i = 1..n)$, i és definida negativa sii $(-1)^i\Delta_i > 0$ $(i = 1..n)$.

Un altre procediment és calcular els valors propis de A (recordem que són tots reals). El signe dels valors propis determina el caràcter de la matriu: si són tots estrictament positius, és definida positiva; si són tots estrictament negatius, és definida negativa; i si n'hi ha dels dos tipus, és indefinida. En el cas que tots tinguin el mateix signe, però alguns siguin zero, la matriu és semidefinida.

6.2 Comandes Maple

Farem servir les comandes `grad` i `hessian` que hem introduït en la secció sobre diferenciabilitat. A més, farem servir noves comandes de tractament de matrius, i de solució de sistemes d'equacions.

La llibreria `linalg` conté moltes funcions per treballar amb matrius, i en destaquem les següents:

`det(`A`);`

retorna el determinant de la matriu quadrada A.

`eigenvalues(`A`);`

retorna la seqüència de valors propis de la matriu A.

`minor(`A`,`i`,`j`);`

retorna el menor de A obtingut eliminant-ne la fila i-èsima i la columna j-èsima.

Per a obtenir punts crítics caldrà resoldre sistemes d'equacions. El Maple té una funció molt poderosa que intenta resoldre sistemes d'equacions en una o diverses variables. La comanda:

`solve(`*eqns*`,`*vars*`);`

tracta d'expressar les solucions del conjunt d'equacions *eqns*, en funció del conjunt de variables *vars*. Retorna un conjunt d'equacions equivalent a *eqns*, on les variables de *vars* apareixen isolades (suposadament; de vegades fracassa).

La comanda `fsolve` és similar a l'anterior, però resol les equacions numèricament; cal, però, que hi hagi tantes equacions com incògnites, i especificar el domini on es cerca una solució.

Finalment, podrem tractar d'obtenir gràficament els punts crítics de funcions de dues variables mitjançant la comanda `implicitplot`.

6.3 Tutorial

6.3.1 Un exemple

Suposem que volem estudiar els màxims i els mínims de la funció $f(x,y) = (x^2+3y^2)e^{1-x^2-y^2}$. Podem dibuixar-ne la gràfica:

```
> with(plots): with(linalg):
> f:=(x,y)->(x^2+3*y^2)*exp(1-x^2-y^2);
> plot3d(f,-2..2,-2..2,axes=boxed);
```

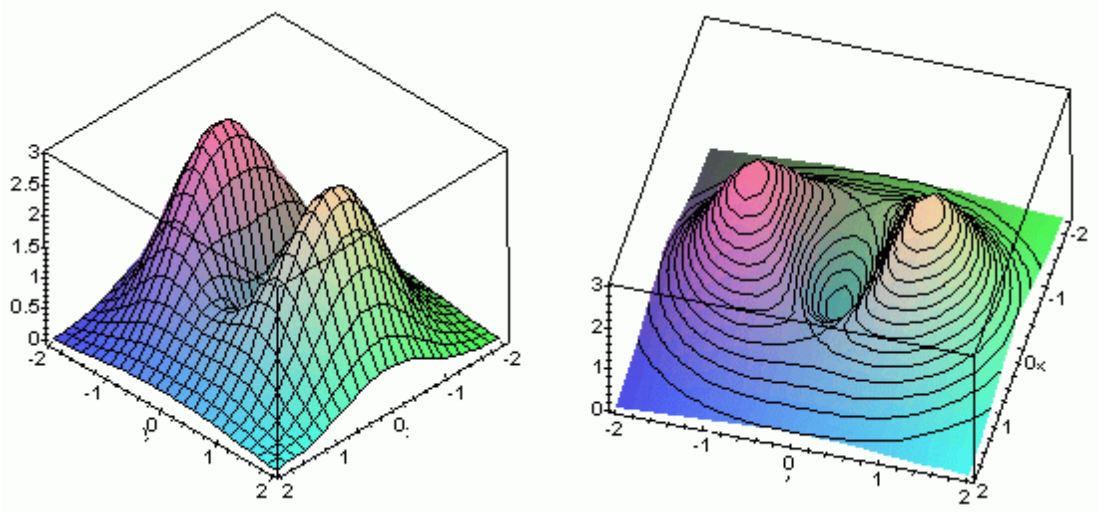

Figura 6.1: Dues visions de la gràfica de $f(x,y) = (x^2+3y^2)e^{1-x^2-y^2}$

L'opció `patchcontour` ens pot donar una millor visió de la gràfica. Hi podem veure que la funció te una parell de màxims, un mínim i dos punts de sella. Geomètricament, els punts de sella són els punts on les corbes de nivell es tallen; i els màxims i mínims són els punts on aquestes "corbes" són un sol punt. Anem ara a obtenir aquest resultat analíticament.

El primer pas consisteix a obtenir els punts crítics de f. Primerament en calculem la matriu jacobiana. Podem definir-la com a funció:

```
> Jf:=(x,y)->subs({X=x,Y=y},grad(f(X,Y),[X,Y]));
```

Tot seguit calculem els punts crítics de f. A tal fi igualem a zero els components d'aquesta matriu, i resolem el sistema corresponent. Gràficament, els punts crítics són els de la intersecció de les corbes obtingudes igualant a zero les derivades parcials:

```
> pl_f1:=implicitplot(Jf(x,y)[1],x=-2..2,y=-2..2,color=blue):
> pl_f2:=implicitplot(Jf(x,y)[2],x=-2..2,y=-2..2,color=red):
> display({pl_f1,pl_f2},axes=none);
```

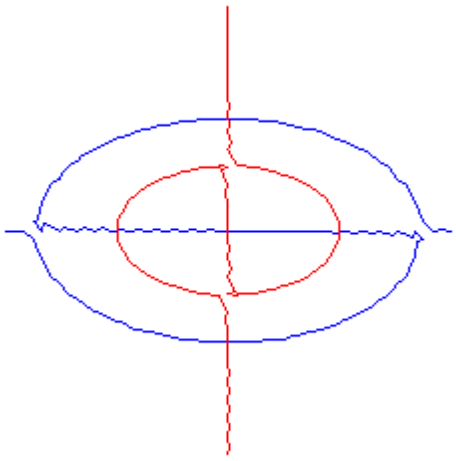

Figura 6.2: Obtenció gràfica dels punts crítics de f

La gràfica resultant (figura 6.2) ens mostra cinc punts crítics dins el rectangle $]-2,2[\times]-2,2[$. (Per cert, n'hem suprimit els eixos, ja que dificultaven la visió de les corbes.)

Podem obtenir els punts mitjançant altres procediments; per exemple, la comanda `solve` resol el sistema $\mathrm{D}f(x,y)=0$:

```
> solve({Jf(x,y)[1]=0,Jf(x,y)[2]=0},{x,y});
```

que dóna com a solucions $(0,0)$, $(\pm 1,0)$, $(0,\pm 1)$. Per tant, no hi ha més punts crítics que els que hem vist.

El segon i darrer pas és estudiar el caràcter dels punts crítics. Només cal calcular la matriu hessiana de f en aquests punts. Definint-la com a funció,

```
> Hf:=(x,y)->subs({X=x,Y=y},hessian(f(X,Y),[X,Y]));
> Hf(0,0); Hf(1,0); Hf(-1,0); Hf(0,1); Hf(0,-1);
```

El resultat són cinc matrius diagonals:

$$\begin{pmatrix} 2e & 0 \\ 0 & 6e \end{pmatrix}, \begin{pmatrix} -4 & 0 \\ 0 & 4 \end{pmatrix}, \begin{pmatrix} -4 & 0 \\ 0 & 4 \end{pmatrix}, \begin{pmatrix} -4 & 0 \\ 0 & -12 \end{pmatrix}, \begin{pmatrix} -4 & 0 \\ 0 & -12 \end{pmatrix},$$

per tant a simple vista sabem el seu caràcter: són, respectivament, definida positiva, dues indefinides, i dues definides negatives. D'aquí el caràcter del punts crítics: mínim, dos punts de sella i dos màxims.

6.3.2 Un altre exemple

Veurem que les coses poden ser més complicades. Considerem la funció $g(x,y)=x^4+y^4-2x^2+4xy-2y^2$. Repetint l'estudi anterior obtenim els gràfics de la figura 6.3, on s'observen tres punts crítics.

Resolem l'equació dels punts crítics:

```
> ppcc := solve({Jg(x,y)[1]=0,Jg(x,y)[2]=0},{x,y});
```

Aquesta crida a `solve` dóna cinc solucions, de les quals les tres primeres són la mateixa: $(0,0)$. Les altres dues contenen l'expressió `RootOf`, i es poden estudiar amb la comanda `allvalues`:

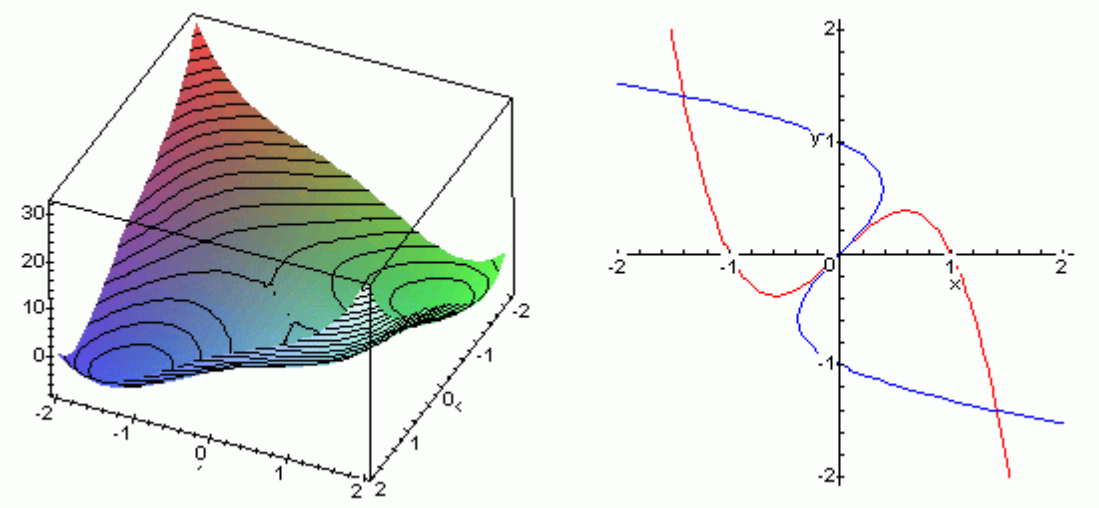

Figura 6.3: Graf de g i obtenció dels seus punts crítics

```
> allvalues(ppcc[4]);
```

dóna dos punts crítics, $\pm(\sqrt{2}, -\sqrt{2})$, mentre que `allvalues(ppcc[5])` dóna quatre solucions, cap de les quals no és real.

Finalment calculem la hessiana de g en els tres punts crítics. Resulta

$$\mathrm{H}g(0,0) = \begin{pmatrix} -4 & 4 \\ 4 & -4 \end{pmatrix}, \quad \mathrm{H}g(\sqrt{2}, -\sqrt{2}) = \mathrm{H}g(-\sqrt{2}, \sqrt{2}) = \begin{pmatrix} 20 & 4 \\ 4 & 20 \end{pmatrix}.$$

Com que $20 > 0$ i $384 > 0$, la darrera matriu és definida positiva, i doncs correspon a dos mínims locals de g. En canvi, la primera matriu és degenerada (té determinant 0), més concretament semidefinida negativa, i per tant a partir d'ella *no* es pot decidir el caràcter del punt crític $(0,0)$. Cal recórrer a altres procediments: observant atentament la gràfica de g al voltant del punt veiem que és un punt de sella. També podem provar d'estudiar la funció g sobre rectes o corbes que passin per l'origen; de fet, $g(x,x)$ té un mínim estricte a $x = 0$, mentre que $g(x,-x)$ hi té un màxim estricte, la qual cosa es pot comprovar, per exemple, fent un `plot` (figura 6.4).

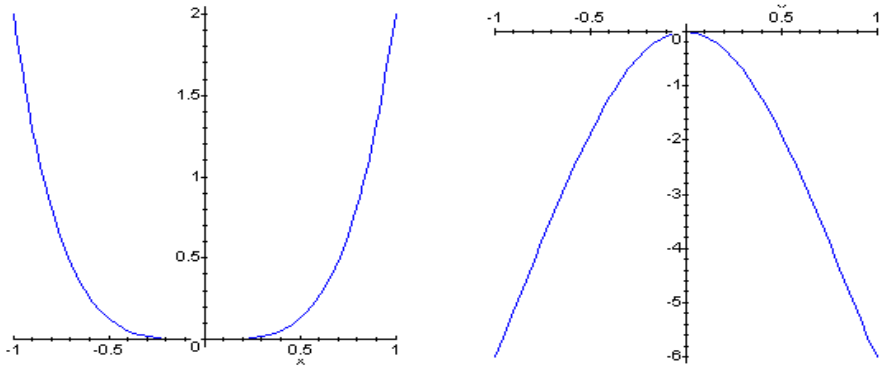

Figura 6.4: Grafs de $g(x,x)$ i $g(x,-x)$

7 Extrems locals condicionats

7.1 Introducció

Els procediments explicats en la secció anterior són útils per a trobar extrems locals de funcions $f\colon V \to \mathbf{R}$ definides en un obert $V \subset \mathbf{R}^n$. Tanmateix, molt sovint interessa estudiar la funció f definida sobre un cert subconjunt $S \subset V$, que pot ser bastant arbitrari. Això passa, per exemple, quan les variables de f no són totes elles independents, sinó que estan lligades (o condicionades) per certes relacions.

En qualsevol cas, la idea de mínim o màxim local, o extrem local, és la mateixa, però el procediment de càlcul serà més complicat. Quin procediment es pot usar? Depèn de com tinguem definit el subconjunt S.

Per exemple, suposem que $S = g(U)$, on $g\colon U \to \mathbf{R}^n$ és una parametrització contínua de S. La funció $\tilde{f}(u) = f(g(u))$ és l'expressió de f sobre S en termes dels paràmetres. Llavors, si $g(u_0) = x_0$ i $f|_S$ té un extrem local en x_0, $\tilde{f}$ té un extrem local en u_0. Per tant mirant els extrems locals de $\tilde{f}$ obtindrem possibles extrems locals de $f|_S$.

El *mètode dels multiplicadors de Lagrange* s'aplica quan S està definit implícitament per $p < n$ equacions $G^1(x) = 0, \ldots, G^p(x) = 0$, on les funcions $G^k\colon V \to \mathbf{R}$ són de classe C^1 i les seves diferencials $\mathrm{D}G^k(x_0)$ són linealment independents. (És a dir, la matriu jacobiana $\mathrm{D}G(x_0) = \left(\partial G^k(x_0)/\partial x^j\right)$ té rang màxim p; aquesta és la condició que assegura que S sigui una subvarietat (corba, superfície, ...) regular de $\mathbf{R}^n$.) En tal cas, si f és diferenciable en x_0 i $f|_S$ té un extrem local en x_0, llavors $\mathrm{D}f(x_0)$ és combinació lineal de les $\mathrm{D}G^k(x_0)$.

En termes més pràctics, cal resoldre el sistema de $n + p$ equacions

$$\left\{\begin{array}{rcl} G^1(x) & = & 0 \\ & \vdots & \\ G^p(x) & = & 0 \\ \partial f(x)/\partial x^1 & = & \lambda_1\, \partial G^1/\partial x^1(x) + \ldots \lambda_p\, \partial G^p/\partial x^1(x) \\ & \vdots & \\ \partial f(x)/\partial x^n & = & \lambda_1\, \partial G^1/\partial x^n(x) + \ldots \lambda_p\, \partial G^p/\partial x^n(x) \end{array}\right.$$

respecte a les $n+p$ variables $(x^1, \ldots, x^n; \lambda_1, \ldots, \lambda_p)$. Les variables λ_k es diuen *multiplicadors de Lagrange*. Un cop resolt el sistema, els punts x obtinguts són els punts crítics de $f|_S$, i són possibles extrems locals.

En algun cas senzill el mètode dels multiplicadors de Lagrange té una interpretació geomètrica. Per exemple, si $f\colon \mathbf{R}^3 \to \mathbf{R}$, i S és la superfície definida per l'equació $G(x) = 0$, on $G\colon \mathbf{R}^3 \to \mathbf{R}$, la condició de ser $x_0 \in S$ un punt crític de f sobre S equival a dir que $\operatorname{grad} f(x_0) = \lambda \operatorname{grad} G(x_0)$, és a dir, la superfície de nivell de f que conté x_0 és tangent a S en x_0.

7.2 Comandes Maple

Les comandes necessàries per a portar a terme el mètode dels multiplicadors de Lagrange ja han estat presentades en seccions anteriors. Afegirem només que Maple en té una que fa automàticament tot el procés. Aquesta funció es diu `extrema` i per a utilitzar-la s'ha de carregar la llibreria del mateix nom.

7.3 Tutorial

Usarem el mètode dels multiplicadors de Lagrange per a trobar el màxim i el mínim de la funció $f(x, y) = x^2 + 4y^3$ sotmesa a la condició $x^2 + 2y^2 = 1$.

```
> with(plots): with(linalg):
> f:=(x,y)->x^2+4*y^3;
> g:=(x,y)->x^2+2*y^2-1;
```

Observem que en aquest cas el lligam entre les variables és l'equació d'una el·lipse C. Per tant, podem parametritzar-la fàcilment:

$$x = \cos(t), y = \frac{\sin(t)}{\sqrt{2}} \quad (t \in [0, 2\pi]).$$

Això ens permetrà de visualitzar els valors de f sobre C:

```
> c:=t->(cos(t),sin(t)/sqrt(2));
> a1:=spacecurve([c(t),0],t=0..2*Pi):
> a2:=spacecurve([c(t),f(c(t))],t=0..2*Pi):
> display(a1,a2,axes=boxed,orientation=[-30,80]);
```

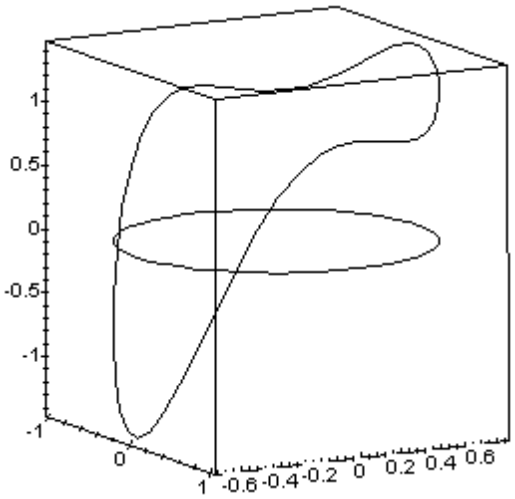

Figura 7.1: Valors de $f(x, y) = x^2 + 4y^3$ sobre l'el·lipse d'equació $x^2 + 2y^2 = 1$

El resultat és a la figura 7.1. Alternativament, podem avaluar f al llarg de la parametrització (figura 7.2):

```
> plot(f(c(t)),t=0..2*Pi);
```

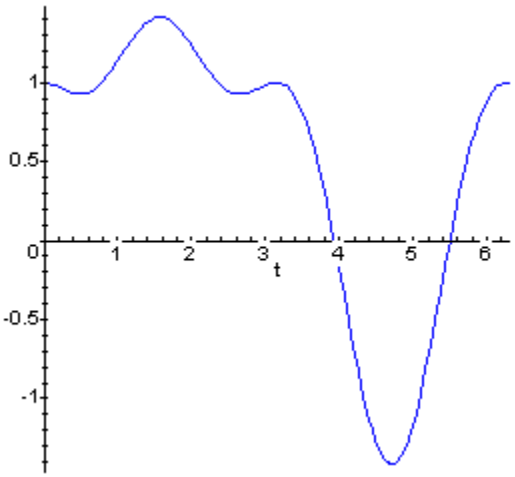

Figura 7.2: Valors de f al llarg de la parametrització c de l'el·lipse

En qualsevol cas s'observen clarament diversos extrems locals.

Tot seguit calcularem aquests extrems mitjançant el mètode dels multiplicadors de Lagrange:

```
> Df:=(x,y)->subs({X=x,Y=y},grad(f(X,Y),[X,Y]));
> Dg:=(x,y)->subs({X=x,Y=y},grad(g(X,Y),[X,Y]));
> solve({Df(x,y)[1]+lambda*Dg(x,y)[1],Df(x,y)[2]+lambda*Dg(x,y)[2],g(x,y)},
  {x,y,lambda});
```

El resultat són quatre solucions:

```
{lambda = -3*RootOf(-1+2*_Z^2), x = 0, y = RootOf(-1+2*_Z^2)},
{y = 0, lambda = -1, x = 1}, {y = 0, lambda = -1, x = -1},
{lambda = -1, y = 1/3, x = 1/3*RootOf(_Z^2-7)}
```

Veiem que vénen donades en funció de les arrels d'uns determinats polinomis. Com que volem trobar les solucions explícitament haurem d'utilitzar la instrucció `allvalues`. Podem procedir així:

```
> solucions:=%;
> sol:=seq(allvalues(solucions[i]),i=1..4);
> for i from 1 to 6 do pc[i]:=op(subs(sol[i],[x,y])) od;
> for i from 1 to 6 do print([pc[i]], f(pc[i]), evalf(f(pc[i]))) od;
```

(Hem utilitzat la comanda `op` per a suprimir els claudàtors dels parells `[x,y]`, i així poder-los avaluar amb `f`. La comanda `print` permet imprimir els resultats en una mateixa línia.) El resultat són sis punts crítics (x, y), acompanyats del valor de $f(x, y)$ i una aproximació decimal d'aquest:

$$\left[0, \frac{1}{2}\sqrt{2}\right], \sqrt{2}, 1.414213562 \qquad \left[0, -\frac{1}{2}\sqrt{2}\right], -\sqrt{2}, -1.414213562 \qquad [1, 0], 1, 1.$$

$$[-1, 0], 1, 1. \qquad \left[\frac{1}{3}\sqrt{7}, \frac{1}{3}\right], \frac{25}{27}, .9259259259 \qquad \left[-\frac{1}{3}\sqrt{7}, \frac{1}{3}\right], \frac{25}{27}, .9259259259.$$

Fixem-nos que C és un conjunt compacte, i per tant $f|_C$ té extrems absoluts. Aquests extrems absoluts corresponen doncs als punts $(0, \pm 1/\sqrt{2})$, on f val $\pm\sqrt{2}$.

Finalment observarem el significat geomètric dels punts crítics. Podem dibuixar simultàniament les corbes de nivell de f que passen pels punts crítics, juntament amb la corba C:

```
> vvcc:={1,sqrt(2),-sqrt(2),25/27};
> plotf:=implicitplot({seq(f(x,y)=z,z=vvcc)},x=-1.2..1.2,y=-1..1,color=red):
> plotg:=implicitplot(g(x,y)=0,x=-1.2..1.2,y=-1..1,color=black):
> plotppcc:=plot({seq([pc[i]],i=1..6)},color=blue,style=point,symbol=circle):
> display({plotf,plotg,plotppcc},scaling=constrained,axes=boxed);
```

(La comanda `seq` construeix una *sequence* amb les quatre expressions o els sis punts que volem dibuixar.) La figura 7.3 mostra la tangència de les corbes de nivell de f amb la corba C sobre els punts crítics:

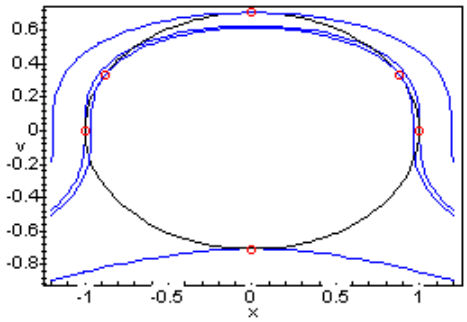

Figura 7.3: La corba C, els punts crítics de $f|_C$, i les corbes de nivell de f

8 Integració múltiple

8.1 Introducció

A vegades, el càlcul d'integrals múltiples es pot fer utilitzant el teorema de Fubini i el teorema del canvi de variables, en termes de primitives de funcions conegudes. Però tot sovint això no és possible. A vegades la situació és pitjor, i no tenim ni tan sols una *expressió* de la funció que volem integrar, l'únic que sabem fer amb ella és avaluar-la en punts. Fins i tot pot passar que només tinguem els valors de la funció en uns quants punts donats i haguem d'aventurar el valor de la integral de la manera més convincent possible. L'objectiu de la integració numèrica és resoldre aquest problema: aproximar tant com es pugui una integral fent la mínima feina possible.

Més concretament, volem calcular la integral

$$\int_U f(x,y)\,\mathrm{d}x\mathrm{d}y,$$

on f és una funció (contínua, per exemple) definida sobre un rectangle compacte $U \subset \mathbf{R}^2$. La definició de la integral de Riemann suggereix de seguir el procediment següent: es divideix el rectangle U en una quadrícula, es calcula el valor de la funció en un punt triat dins de cada quadradet, multiplicat per l'àrea del quadradet, i se sumen els resultats.

Sembla raonable esperar que com més fina sigui la partició en quadradets, més ens aproximarem al valor exacte de la integral. Doncs és així: es pot demostrar que aquest procediment, conegut com la *regla dels rectangles composta*, convergeix efectivament, a mesura que la partició es fa més fina, cap a la integral de la funció.

Tanmateix, aquesta no és la millor manera de fer-ho. La convergència és massa lenta. Què hauríem de canviar? La idea de multiplicar el valor de la funció per l'àrea del quadradet corresponent potser no sigui tan bona com sembla. Estem donant la mateixa importància als punts de la vora de U que als punts centrals. Poden inventar-se regles d'integració que donin un "pes" diferent a cada punt, segons la seva posició dins de U. Si això es fa amb una certa gràcia, s'obtenen regles amb molt bones propietats. Més endavant en veurem una.

8.2 Comandes Maple

Abans de plantejar-nos la integració numèrica, recordem què és capaç de fer el Maple. La comanda `int` calcula integrals definides i indefinides d'expressions d'una variable. La sintaxi és la següent:

`int(`*expr*`,`*x*`);`

calcula la primitiva de l'expressió *expr* en la variable x (amb la constant d'integració més còmoda possible).

Si s'especifica un interval d'integració:

`int(`*expr*`,`*x*`=`*a*`..`*b*`);`

llavors es retorna el valor de la integral definida entre a i b. Encara que el Maple no pugui calcular simbòlicament alguna integral definida, la comanda `evalf` en donarà una aproximació numèrica. Aplicant `int` dues o tres vegades, gràcies al teorema de Fubini, poden calcular-se integrals dobles o triples.

En aquesta secció farem ús de *procediments* de Maple. No explicarem detalladament el seu funcionament, però és força intuïtiu i no costa gaire habituar-s'hi.

8.3 Tutorial

8.3.1 Un exemple

Calcularem la integral de la funció $f(x, y) = y^2 - x^3$ sobre el rectangle $[-1, 1] \times [-1, 1]$. Podem definir-la i dibuixar-ne la gràfica:

```
> f := (x,y) -> y^2-x^3;
> plot3d(f(x,y),x=-1..1,y=-1..1,axes=framed);
> int(int(f(x,y),x=-1..1),y=-1..1);
```

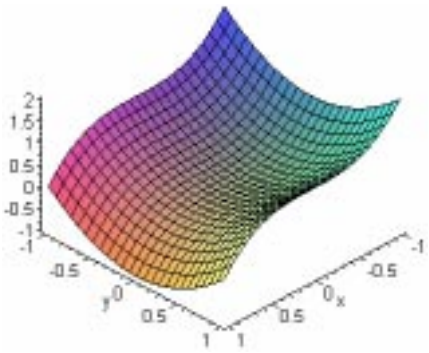

Figura 8.1: Gràfica de la funció que volem integrar

El resultat de la integració doble és 4/3. El Maple no ha tingut cap problema a obtenir-lo, ja que f és un polinomi. Podem utlitzar aquest exemple com a test dels procediments numèrics que estudiarem.

8.3.2 La regla dels rectangles composta

Anem a dissenyar un procediment de Maple per a la regla dels rectangles composta. El procediment, que anomenarem `rectangles2d`, rep com a entrada una funció f, el rectangle $[a, b] \times [c, d]$ on la volem integrar, i el nombre de divisions per fer a cada costat del rectangle. Per simplificar, suposarem que dividim el rectangle en una quadrícula de n^2 rectangles iguals. Primer escrivim el procediment i després el comentem:

```
> rectangles2d := proc(f,a,b,c,d,n)
>         local x, y, s;
>         s := 0;
>         for x from a+(b-a)/(2*n) by (b-a)/n while x < b do
>                 for y from c+(d-c)/(2*n) by (d-c)/n while y < d do
>                         s := s + f(x,y);
>                 od;
>         od;
>         s := s*(b-a)*(d-c)/(n^2);
>         eval(s);
> end;
```

Aquest procediment pot semblar una mica llarg, però no fa res d'especial. La primera línia conté les dades d'entrada que ja hem especificat. Es fan servir tres variables: la `x` i la `y` porten el control del punt on estem, i la `s`, inicialment zero, va acumulant la suma de la funció avaluada en tots els punts.

Recordem que en la regla dels rectangles composta, avaluem la funció en un punt de cada rectangle. En aquest cas hem pres el seu centre. El centre del primer rectangle té coordenades

$$(x, y) = \left(a + \frac{b-a}{2n},\ c + \frac{d-c}{2n}\right)$$

i la distància entre els centres de dos rectangles veïns és $(b-a)/n$ en horitzontal, i $(d-c)/n$ en vertical (vegeu la figura 8.2).

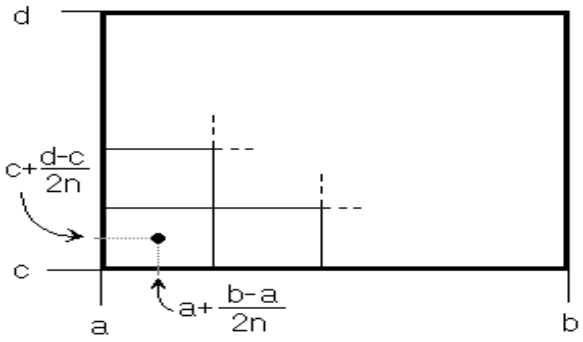

Figura 8.2: Partició en n^2 rectangles

Finalment, com que tots els rectangles petits tenen la mateixa àrea ($(b-a)(d-c)/n^2$), no cal que anem multiplicant cada $f(x,y)$ per aquest nombre, sinó que ho multipliquem tot plegat al final. Un cop explicat, anem a provar com funciona.

Si fem

```
> n := 1;
> rectangles2d(f,-1,1,-1,1,n);
```

el resultat és 0, que és el mateix que $f(0,0)$. Sembla que funcioni bé, ja que la regla dels rectangles composta amb $n=1$ és equivalent a avaluar la funció en el centre del rectangle (i multiplicar per la seva àrea). Si fem $n=30$, obtenim un resultat proper a $4/3 = 1.333\ldots$. Per a veure a quina velocitat ens aproximem a la solució en incrementar n, podem fer

```
> for n from 1 to 50 do evalf(rectangles2d(f,-1,1,-1,1,n)); od;
```

N'obtenim una llista de 50 nombres:

$$0,\ 1,\ 1.185,\ 1.25,\quad \ldots\quad 1.33185$$

Pot semblar que ens estiguem aproximant al valor correcte, però pensem que hem avaluat la funció en $50^2 = 27500$ punts, i només hem obtingut tres xifres significatives correctes.

8.3.3 La regla de Simpson

Fixem-nos que l'única cosa que hem fet ha estat avaluar la funció en uns quants punts, i fer una suma ponderada dels resultats (en aquest cas, tots amb el mateix pes). Però ja hem dit en la introducció que hi ha maneres més enginyoses d'escollir el pes de cada punt. Pot sorprendre, però si avaluem la funció en 9 punts, el millor que podem fer és dotar-los dels pesos següents:

$$\begin{matrix} 1 & 4 & 1 \\ 4 & 16 & 4 \\ 1 & 4 & 1 \end{matrix}$$

cada un d'ells multiplicat per $\frac{(b-a)(d-c)}{36}$. Aquesta disposició de pesos té nom propi: és la *regla de Simpson bidimensional*, i està basat en la regla de Simpson per al càlcul d'integrals en una variable.

Per a veure com funciona, primerament escriurem un procediment que la realitzi:

```
> simpson2d := proc(f,a,b,c,d)
> local s;
>       s := 0;
>       s := s + f(a,c);
>       s := s + 4*f(a+(b-a)/2,c);
>       s := s + f(b,c);
>       s := s + 4*f(a,c+(d-c)/2);
>       s := s + 16*f(a+(b-a)/2,c+(d-c)/2);
>       s := s + 4*f(b,c+(d-c)/2);
>       s := s + f(a,d);
>       s := s + 4*f(a+(b-a)/2,d);
>       s := s + f(b,d);
>       eval(s*(b-a)*(d-c)/36);
> end;
```

Observem que, a aquest procediment, no li passem cap nombre n, ja que —en principi— sempre avalua la funció en exactament 9 punts. Si ho provem amb la funció $f(x, y) = y^2 - x^3$ sobre el rectangle $[-1, 1] \times [-1, 1]$,

```
> simpson2d(f,-1,1-1,1);
```

dóna com a resultat $4/3$. El valor exacte.

De fet, la regla de Simpson calcula exactament integrals de polinomis on no apareguin potències més grans que 3. Per exemple, també integrarà exactament la funció $g(x, y) = 3 - 7xy^2 + 4x^3y^2 - 9x^3 + 2y^3x^3 + 5y$ sobre qualsevol rectangle. En realitat, la regla de Simpson va ser inventada amb aquest fi: si hom es demana quins pesos cal posar per tal que s'integrin exactament tots els polinomis de grau ≤ 3, la resposta ve donada per la regla de Simpson.

8.3.4 Un altre exemple

Anem a veure que, de fet, la regla de Simpson funciona força bé per a funcions qualssevol, sempre que els rectangles siguin petits. Per exemple, integrem la funció $f(x, y) = \sin(xy)$ sobre el rectangle $[0, 1] \times [0, 1]$:

```
> f := (x,y) -> sin(x*y);
> evalf(int(int(f(x,y),x=0..1),y=0..1));
```

dóna com a resultat $0.2398117420\ldots$. Prenem aquest resultat com a bo. L'aplicació de `simpson2d` dóna $0.2398705178\ldots$; no està gens malament, avaluant la funció en 9 punts obtenim 4 decimals correctes. En canvi, aplicant `rectangles2d` amb 9 punts obtenim un sol decimal correcte; per tal d'aconseguir-ne quatre necessitem $n = 15$, que són 225 punts.

Per tant, sembla que Simpson és una bona elecció enfront de la regla dels rectangles. Un cop vist això, no és difícil inventar-se una *regla de Simpson composta*, on es divideix el rectangle original en quadradets, i en cada un d'ells s'aplica la regla de Simpson. Fixem-nos que els quadradets veïns tindran punts comuns, i per tant es comptaran els seus pesos per duplicat (o quadruplicat). Per exemple, en una descomposició en 3×2 quadradets tindríem els pesos següents:

$$\begin{array}{ccccccc} 1 & 4 & 2 & 4 & 2 & 4 & 1 \\ 4 & 16 & 8 & 16 & 8 & 16 & 4 \\ 2 & 8 & 4 & 8 & 4 & 8 & 2 \\ 4 & 16 & 8 & 16 & 8 & 16 & 4 \\ 1 & 4 & 2 & 4 & 2 & 4 & 1 \end{array}$$

Així és com es calculen les integrals en la vida real, amb la regla de Simpson composta. Aquest mètode es generalitza fàcilment a dimensions més grans que 2, i funciona força bé per dimensions petites (fins a 7 o 8).

9 Integrals de línia i de superfície

9.1 Introducció

Les integrals de línia i de superfície tenen nombroses aplicacions físiques. Aquí només considerarem integrals de funcions vectorials. En aquest cas, les integrals de línia es diuen *circulacions* i les integrals de superfície es diuen *fluxos*. La integració de funcions escalars és més senzilla, i el seu càlcul amb Maple es pot deduir del que veurem en aquesta secció.

Per a calcular una circulació, les dades són un camp vectorial F i una corba orientada C. La corba C usualment ve donada per una parametrització $\sigma\colon I \to \mathbf{R}^3$, on $I \subset \mathbf{R}$ és un interval. La definició de la circulació és llavors

$$\int_C F \cdot \mathrm{d}l := \int_I F(\sigma(t)) \cdot \sigma'(t)\,\mathrm{d}t,$$

on $\sigma'(t)$ és el vector tangent de σ a l'instant t.

Per a calcular un flux, les dades són un camp vectorial F i una superfície orientada S. La superfície S tot sovint ve donada per una parametrització $g\colon U \to \mathbf{R}^3$, on $U \subset \mathbf{R}^2$ és un obert. La definició del flux és similar a la de circulació, però ara, el paper que abans feia el vector tangent, el fa el vector normal definit per g:

$$\int_S F \cdot \mathrm{d}S := \int_U F(g(u,v)) \cdot N(u,v)\,\mathrm{d}u\mathrm{d}v,$$

on $N = T_1 \times T_2$ i els T_i són els vectors tangents definits per la parametrització,

$$T_1(u,v) = \frac{\partial g}{\partial u}(u,v), \quad T_2(u,v) = \frac{\partial g}{\partial v}(u,v).$$

Finalment, recordem que, sota certes hipòtesis, el teorema de Stokes relaciona aquestes dues integrals:

$$\int_S \operatorname{rot} F \cdot \mathrm{d}S = \int_{\partial S} F \cdot \mathrm{d}l,$$

on ∂S és la vora de S amb l'orientació induïda (figura 9.1).

Figura 9.1: Una bona elecció i una mala elecció de les orientacions

Anàlogament, el teorema de Gauss afirma que

$$\int_V \operatorname{div} F\,\mathrm{d}V = \int_{\partial V} F \cdot \mathrm{d}S,$$

on $V \subset \mathbf{R}^3$ és un obert i ∂V la seva vora, orientada vers l'exterior.

9.2 Comandes Maple

Utilitzarem els operadors diferencials `diff`, `grad` i `jacobian`, que ja hem presentat prèviament. També hi ha

`curl(`*exprs*`,`*vars*`);`

que calcula el rotacional, i

`diverge(`*exprs*`,`*vars*`);`

que calcula la divergència.

També requerirem algunes comandes de manipulació de vectors. El producte escalar de dos vectors *vect1* i *vect2* de la mateixa dimensió es calcula amb

`innerprod(`*vect1*`,`*vect2*`);`

Si a més *vect1* i *vect2* són de dimensió 3, podem fer-ne el producte vectorial:

`crossprod(`*vect1*`,`*vect2*`);`

Una altra comanda que ens serà útil és `col`, que permet extreure columnes d'una matriu. Així,

`col(`A`,`i`);`

retorna la columna i-èsima de la matriu A, en forma de vector.

Totes aquestes comandes s'han de carregar de la llibreria `linalg`.

9.3 Tutorial

Per il·lustrar com poden fer-se integrals de línia i de superfície amb Maple, comprovarem que es compleix el teorema de Stokes en un cas senzill. Això ens obligarà a fer una integral de línia (una circulació) i una integral de superfície (un flux).

Concretament, considerem el camp vectorial i la superfície donats per

$$F(x,y,z) = (z,x,y) \qquad S = \left\{(x,y,z) \in \mathbf{R}^3 \mid x^2+y^2+z^2 = 1,\ z < 0\right\}.$$

S és l'hemisferi sud de l'esfera de radi 1 amb centre a l'origen; l'orientem (per exemple) amb la normal "cap avall". Volem calcular les integrals $\displaystyle\int_S \operatorname{rot} F \cdot \mathrm{d}S = \int_{\partial S} F \cdot \mathrm{d}l$.

Primer de tot, parametritzem S. Amb aquest fi podem usar com a paràmetres les variables angulars (θ, ϕ) de les coordenades esfèriques:

```
> g := (theta,phi) -> [sin(theta)*cos(phi),sin(theta)*sin(phi),cos(theta)];
> rang_theta := theta=Pi/2..Pi;
> rang_phi := phi=0..2*Pi;
```

També parametritzem la vora de S:

```
> c := t -> g(Pi/2,t);
> rang_t := t=0..2*Pi;
```

Hem definit els rangs perquè els haurem de fer servir més d'un cop: per fer els dibuixos i per calcular les integrals. Podem dibuixar la superfície i destacar-ne la vora de la manera següent:

```
> superficie := plot3d(g(theta,phi),rang_theta,rang_phi,
>         style=hidden,color=coral):
> vora := tubeplot(c(t),rang_t,radius=0.01,color=black):
> display(superficie,vora);
```

Examinarem també si les orientacions definides per g i c són les corresponents a S i la seva vora. A tal fi calculem els vectors tangent definit per c i normal definit per g. El darrer es pot calcular amb la instrucció `jacobian`:

```
> Tc := t -> subs(T=t,diff(c(T),T)):
> Tc(t);
> Jg := jacobian(g(U,V),[U,V]);
> T1 := col(Jg,1): T2 := col(Jg,2):
> N := (theta,phi)->subs({U=theta,V=phi},simplify(crossprod(T1,T2))):
> N(theta,phi);
```

Podem prendre punts concrets de la superfície i de la corba i examinar cap a on apunten aquests vectors. També podem representar-los gràficament:

```
> angles_o:=(2*Pi/3,0); t_o:=Pi/4;
> fletxac := arrow(c(t_o),Tc(t_o),[1,1,1],0.05,0.2,0.3,color=black):
> fletxag := arrow(g(angles_o),N(angles_o),[1,1,1],0.05,0.2,0.3,color=black):
> display(superficie,vora,fletxac,fletxag,orientation=[45,65]);
```

Aquí hem usat la comanda `arrow`, que requereix la llibreria `plottools`. Aquesta comanda permet dibuixar fletxes en gràfics tridimensionals. Com és lògic, cal donar-li un punt i un vector, així com altres paràmetres addicionals que podeu consultar en el *help*. El resultat és la figura 9.2.

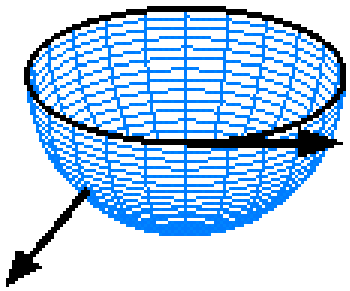

Figura 9.2: La superfície i la seva vora, amb les orientacions donades per les parametritzacions

Hi veiem clarament que l'orientació de S donada per g és l'apropiada, mentre la parametrització c orienta ∂S amb l'orientació oposada.

Necessitem el camp vectorial F i el seu rotacional. Poden definir-se amb les comandes següents:

```
> F := (x,y,z) -> [z,x,y];
> rotF := (x,y,z) -> subs({X=x,Y=y,Z=z},curl(F(X,Y,Z),[X,Y,Z])):
```

Fixem-nos que hem fet servir la comanda `curl`, que treballa amb expressions, i cal fer el truquet de la substitució per tal de tenir una funció. En aquest cas concret, $\operatorname{rot} F$ és el camp constant $(1,1,1)$.

Ara ja podem calcular les integrals. Calculem primer la circulació de F al llarg de la corba ∂S:

```
> int( innerprod(F(op(c(t))),Tc(t)),rang_t );
```

que dóna com a resultat π. Com que l'orientació definida per c és l'oposada de l'orientació de ∂S, la circulació és $-\pi$. Remarquem una cosa sobre aquesta comanda: `op` treu els claudàtors de l'expressió del vector `c(t)` per tal que `F` pugui avaluar-s'hi.

I ara calculem el flux de $\operatorname{rot} F$ a través de S:

```
> int(
>     int(
>         innerprod(rotF(op(g(theta,phi))),N(theta,phi)),
>     rang_theta),
> rang_phi);
```

El resultat és també $-\pi$.

En aquest cas, les integrals són força senzilles i gairebé podrien fer-se a mà. Però si fem exemples una mica més complicats, el Maple no sabrà trobar les primitives simbòlicament (ja que, segurament, no es podran expressar en termes de funcions elementals). Llavors caldrà avaluar-les numèricament, usant la comanda `evalf`, com s'ha explicat en la secció anterior.

Apèndix: expressions i funcions en Maple

És essencial distingir bé entre funcions i expressions a l'hora de programar en Maple. Quan fem matemàtiques amb paper i llapis cometem molts abusos de notació, que són còmodes, però que resultarien difícils de fer entendre a una màquina. Per tant, s'ha d'anar amb compte amb què estem dient al Maple quan escrivim certes coses ...

Per exemple, suposem que volem treballar amb la funció $f\colon \mathbf{R}^2 \to \mathbf{R}$ definida per $f(x,y) = x^2 - 3y^2$. En Maple, hi ha bàsicament dues maneres de començar. Podem definir la funció com a tal:

```
> fun := (x,y) -> x^2 - 3*y^2;
```

O bé escriure simplement la seva expressió

```
> expr := x^2 - 3*y^2;
```

La diferència més important és que els símbols `x` i `y` fan un paper especial en `expr`, mentre que en `fun` només són un artilugi notacional que ens permet de definir la funció. Si féssim servir qualssevol altres variables per a definir `fun`, per exemple `u` i `v`, el resultat seria el mateix. Però en el cas de `expr` seria ben diferent.

Què es pot fer amb una funció? El més fàcil és avaluar-la en un punt. Si fem

```
> fun(1,1);
```

el resultat és -2. No cal avaluar-la en un punt numèric, ho podem fer en un de simbòlic. Així, en fer

```
> fun(x,y);
```

obtenim, exactament, `x^2 - 3*y^2`.

Què es pot fer amb una expressió? Podem substituir les seves variables per nombres, o bé per altres símbols. Per exemple, si fem

```
> subs(x=1,y=1,expr);
```

el resultat és -2.

Els avantatges de les funcions són molts: són fàcils d'avaluar en un punt, són fàcils de compondre les unes amb les altres (penseu com es faria això amb expressions), i no depenen de símbols concrets. En canvi pot semblar que les expressions siguin molt incòmodes. Per exemple, si en algun moment féssim `x:=0`, l'expressió `expr` se n'aniria a norris —es convertiria en `-3*y^2`—, mentre que la funció `fun` quedaria intacta —`fun(x,y)` seguiria donant `x^2 - 3*y^2`.

Malauradament, però, el Maple treballa més fàcilment amb expressions que no pas amb funcions. Hi ha comandes (per exemple, el càlcul del límits) que només es poden cridar amb expressions. Per sort, és molt fàcil de passar de les unes a les altres.

Ja que "aplicar" una funció la transforma en una expressió, no és estrany que l'operació inversa, transformar una expressió en una funció, rebi en Maple el nom de `unapply`. Així doncs, l'operació inversa de fer

```
> expr := fun(x,y);
```

no és res més que fer

```
> fun := unapply(expr,x,y);
```

Seria exagerat dir que, si s'entén bé això, ja es té dominat el 67% de la programació en Maple?

Índex

allvalues, 22, 26
animate, 5
arrow, 33

col, 32
contourplot, 2, 3
crossprod, 32
curl, 31

D, 8
definició de conjunts, 1
definició de la diferencial com a funció, 11
definició de rangs, 1
det, 20
diff, 9
display, 16, 26
diverge, 32

eigenvalues, 20
evalf, 27
expressió, 35

funció, 35

grad, 9

hessian, 9

implicitplot, 2, 3, 13
implicitplot3d, 13
innerprod, 32
int, 27

jacobian, 9, 32

limit, 5

minor, 20
mtaylor, 18

op, 26

plot, 5, 13
plot3d, 1, 13
print, 26
procediment, 28

readlib, 18

seq, 26
setoptions3d, 2
solve, 21
spacecurve, 14
subs, 11

tubeplot, 14, 32

unapply, 35