Felhasználói függvények definiálása és függvények 3D ábrázolása SCILAB programcsomag segítségével

1. Felhasználói függvények definiálása

A Scilab programcsomag rengeteg matematikai függvényt biztosít a számítások elvégzésére, ezek egy része a jól ismert függvény (exp(), sin(), ...), a többi pedig előre elkészített, valamilyen feladat megoldására szolgáló Scilab függvény (inv(), linsolve(), ...).

A felhasználói függvényeket online módon kétféleképpen lehet definiálni, egyrészt egyetlen parancssorban, másrészt több parancssorban. Mindkettő alkalmazására mutatunk példát.

1.1. Függvény definiálása egy parancssorban

Egyszerűbb függvényeket ezzel a módszerrel definiáljuk. A **deff()** nevű Scilab definíciós függvénnyel végezzük, amelynek két paramétere van, mindegyik sztring paraméter. Az első paraméter a függvényhívást, a második paraméter pedig a függvénykiértékelést reprezentálja. A függvénydefiníciót az $y = x^2$ függvény definiálásával mutatjuk be. A függvényünk neve legyen *negyzet*. Az alábbi parancsot használjuk a definiáláshoz

deff('y=negyzet(x)','y=x^2')

A definíciós sorban az "y" és az "x" betüknek nincs jelentősége a későbbiekben, mivel ezek csupán a formula leírására szolgálnak. A "negyzet" viszont fontos, hisz ez a név azonosítja a függvényünket, ezzel a névvel tudunk hivatkozni rá a későbbi felhasználásnál. A függvény hívása úgy történik, hogy leírjuk a nevét és zárójelben megadjuk az argumentumát, például:

```
->negyzet(4)
ans =
16.
```

A felhasználó által létrehozott függvény (hasonlóan az előre definiált Scilab függvényekhez) tud használni vektort is argumentumként. Legyen például az alábbi számsorozat, amely elemeinek a négyzetét egyszerűen kiszámíthatjuk és elhelyezhetjük például az "fn" vektorváltozóba az alábbiak szerint

```
->z=-2:0.5:2
z =
- 2. - 1.5 - 1. - 0.5 0. 0.5 1. 1.5 2.

->fn=negyzet(z)
fn =
4. 2.25 1. 0.25 0. 0.25 1. 2.25 4.
```

Számunkra az optimalizálásban a többváltozós függvények lesznek érdekesek, ezért ezekre is mutatunk példát. Legyen a kétváltozós függvényünk az alábbi: $z = x^4 - y^4$, amelynek neve legyen "fv", megadása a következő módon történhet:

```
->deff('z=fv(x,y)','z=x^4-y^4')
```

A függvény kiszámítása a (2,1) pontban:

```
->fv(2,1)
ans =
15
```

Az előző példában definiált függvénynek két valós változója van. Definiálhatjuk ugyanezt a függvényt egy változóval is, amely természetesen már vektorváltozó lesz. Az alábbi példával illusztráljuk ezt a definiálást, ahol az "x" változó most egy sorvektor vagy egy oszlopvektor:

```
->deff('z=ffv(x)','z=x(1)^4-x(2)^4')

z=[2;1]
z =
2.
1.

->ffv(z)
```

```
ans =
15.

-->x=[2 1]
x =
2. 1.

-->ffv(x)
ans =
15.
```

Az, hogy milyen módon definiáljuk a függvényt, a felhasználó dolga. Ha kettőnél több változó van, akkor célszerű az "f(x)" típusú definiálás. A függvény ábrázolásánál, mivel ott legfeljebb kétváltozós lehet a függvény, az első típusú definiálást kell alkalmazni, a Scilab grafikus beépített függvényei ezt igénylik.

Az alábbiakban a négydimenziós x vektor hosszának a négyzetét definiáljuk. A függvény $f(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2$, neve pedig legyen "ffvv".

```
-->deff('x=ffvv(z)','x=z(1)^2+z(2)^2+z(3)^2+z(4)^2')

-->y=[2;1;-1;3];
-->ffvv(y)
ans =
15.
```

1.2. Függvény definiálása több parancssorban

Ezt a fajta definiálást leginkább akkor alkalmazzuk, ha a függvényünk formulája bonyolult. A definiálást egy keretben végezzük, amelynek alakja a következő:

function y=fnev(x)
...
a függvényt leíró parancsok
...
endfunction

Az első sorban a Scilab **function** alapszava után adjuk meg a függvényhívás reprezentációját. Ez hasonlít az előző definicióhoz. A következő sorokban adjuk meg a függvény kiszámításának módját. Az **endfunction** alapszó zárja le a függvény definícióját. Fontos, hogy az "y" kimenő paraméternek legalább egyszer értéket adjuk a függvényt leíró parancssorokban.

Tekintsük az alábbi egyváltozós függvényt:

$$y = \begin{cases} x & , \text{ha} \quad x \geqq 2 \\ x + (x-2)^2 & , \text{ha} \quad x < 2 \end{cases}$$

A fenti függvény megadása a következő:

```
-->function y=f(x)
-->if x>=2 then y=x; end;
-->if x<2 then y=x+(x-2)^2; end;
-->endfunction

-->f(5)
ans =
5.

-->f(1)
ans =
2.
```

Most tekintsük a következő kétváltozós függvényt. Legyen a függvény értéke az origó közepű 3 sugarú körön és annak belsejében $9-(x_1^2+x_2^2)$, a körön kívül pedig $x_1^2+x_2^2$. Ez valójában két forgási paraboloid, az első lefelé, a második felfelé szélesedik.

$$z=\begin{cases} 9-(x_1^2+x_2^2) & \text{, ha} \quad x_1^2+x_2^2 \leqq 9 \\ x_1^2+x_2^2 & \text{, ha} \quad x_1^2+x_2^2>9 \end{cases}$$

A függvényünk kétváltozós, ahogy az alábbiakban definiáljuk valójában "egyváltozós" lesz, de az a változó vektorváltozó:

```
–>function z=ff(x)
–>if x(1)^2+x(2)^2 <= 9 then z=9-(x(1)^2+x(2)^2); end
–>if x(1)^2+x(2)^2 > 9 then z=x(1)^2+x(2)^2; end
–>endfunction

–>x=[1;2];
–>ff(x)
ans =
4.

–>z=[4;3];
–>ff(z)
ans =
25.
```

Az előző előismeretek után most megadjuk mindkét függvény-definíció általános formáját.

deff('[y1,y2,. . . ,ym]=fvnev(x1,x2,. . . ,xn)','fvképlete')

function [y1,y2,. . . ,ym]=fnev(x1,x2,. . . ,xn)
. . .
a függvényt leíró parancsok
. . .
endfunction

ahol az **fvnev** a függvény neve, az **y1,y2,. . . ,ym** változók a függvény kimenő paraméterei, az **x1,x2,. . . ,xn** változók a függvény bemenő paraméterei. Mindegyik fajta paraméter lehetnek vektor vagy mátrixok is. Tehát a kimenő paraméterek száma az eddig megszokottól eltérően egynél több is lehet. Ha egy kimenő paraméter van, akkor a szögletes zárójel el is maradhat, ahogy eddig is tettük. Ne feledkezzünk meg arról, hogy az első típusú definiálásnál a **deff()** függvénynek mindkét paramétere sztring, tehát aposztrófokat kell használni.

Most tekintsük az $f(x)=x_1^2+x_1x_2+2x_2^2$ függvényt és készítsünk olyan Scilab függvényt, amely a függvényt és annak gradiens vektorát adja kimenő paraméterként, legyen ennek a függvénynek a neve *fvgr*.

```
–>deff('[f,g]=fvgr(x)','f=x(1)^2+x(1)*x(2)+2*x(2)^2,g=[2*x(1)+x(2),x(1)+4*x(2)]')

–>x=[2;-1];

–>[fv,gr]=fvgr(x)
gr =
3. - 2.
fv =
4.
```

Mint tudjuk az argumentuma ennek a függvénynek egy vektor, amely lehet sor vagy oszlopvektor. A definícióban a gradiens vektort sorvektorként definiáltuk, ezért kaptunk sorvektort eredményül. Természetesen lehetett volna a gradiens vektort oszlopvektorként is definiálni, ekkor a függvénydefiniálás és a függvényhívás eredménye az alábbi:

–>deff('[f,g]=fvgro(x)','f=x(1)^2+x(1)*x(2)+2*x(2)^2,g=[2*x(1)+x(2);x(1)+4*x(2)]')

–>[fv,gr]=fvgro(x)
gr =
3.
- 2.
fv =
4.

2. Függvények háromdimenziós (3D) ábrázolása

Az alábbiakban a háromdimenziós (3D) ábrázolás grafikus parancsai közül az alábbi kettőt mutatjuk be:

plot3d(x,y,z)
fplot3d(x,y,fvnev)

Az **x**, **y** egy-egy vektor, amely kijelöli, hogy az (**x**,**y**) síkon mely pontokhoz tartozó függvényértékeket akarjuk megjeleníteni. Ehhez elő kell állítani a **z** mátrixot, amely a pontokbeli függvényértékeket tartalmazzák. Ezzel a paranccsal nem foglalkozunk, mert mi függvényeket szeretnénk ábrázolni, amelyre a második parancs a leginkább alkalmas. A második parancsban harmadik paraméterként (**fvnev**) az ábrázolandó függvény nevét kell megadni. Megjegyezzük, hogy a grafikus parancsoknak több argumentumuk is lehet, ezek tanulmányozását az olvasóra bízzuk.

A 3D függvényábrázoláshoz tehát az alábbi két dolgot kell elvégezni:

1. Az ábrázolandó kétváltozós függvény definiálása.
2. Az **x** és az **y** vektorok előállítása.

Tekintsük az alábbi kétváltozós függvényt. Legyen a függvény értéke az origó közepű 3 sugarú körön és annak belsejében 5, a körön kívül pedig $2x + y$, tehát két síkot kell ábrázolni.

1. lépés: A függvény definiálása

–>function z=fff(x,y)
–>if x^2+y^2 <= 9 then z=5; end
–>if x^2+y^2 > 9 then z=2*x+y; end
–>endfunction

2. lépés: Az **x** és az **y** vektorok megadása

Egy téglalap tartományon akarjuk megjeleníteni a függvényt, a tartomány mindkét irányban -5 és 5 között legyen. A pontokat 0.1 távolságra vesszük fel, így a vektorok megadása a következő:

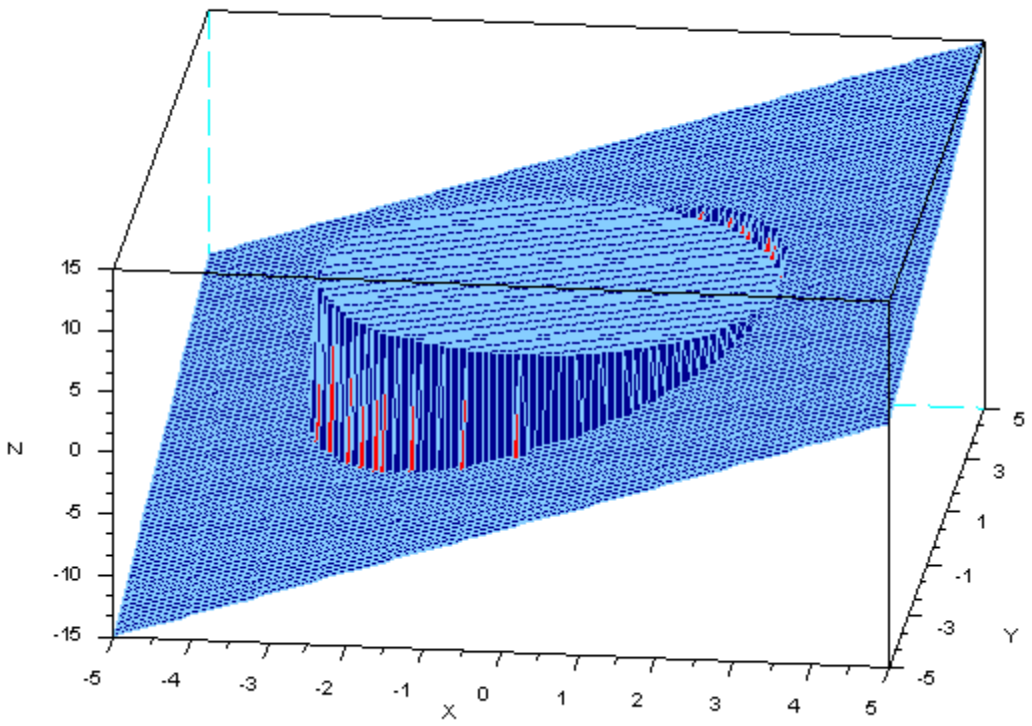
–>x=[-5:0.1:5];
–>y=x;

3. lépés: A függvény megjelenítése

Ehhez ki kell adni a grafikus parancsot:

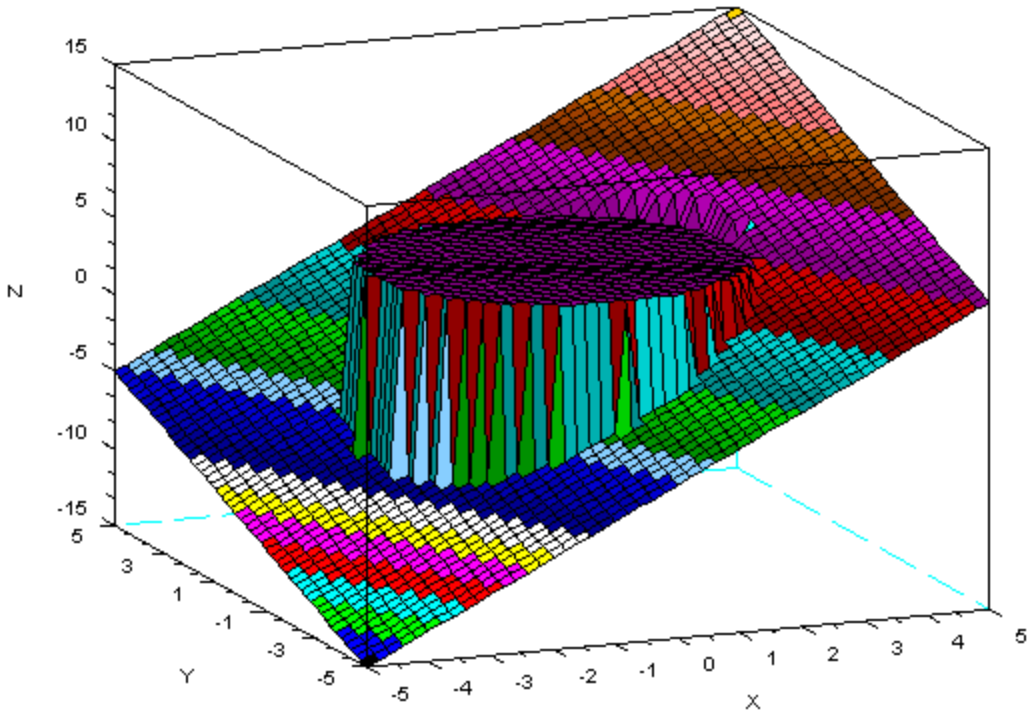
–>fplot3d(x,y,fff)

E parancs kiadása után a függvény képe egy külön ablakban jelenik meg. Ennek a grafikus megjelenítőnek is van menüje, a függvény képének sok paramétere állítható be, amelynek ismertetésére nem térünk ki, jó szórakozást jelenthet az olvasónak ennek áttanulmányozása. Lehetőség van a kép színezésére, forgatására, stb.



A bemutatott grafikus függvényeknek még van egy hasznos változatuk is, ezek a **plot3d1()** és az **fplot3d1()** függvények. Ezek alkalmazásával szemléletesebb képet kaphatunk, mert a függvény kölönböző szintjeit más-más színnel jeleníti meg. Az előző függvény ábrázolása az új grafikus függvénnyel az alábbi parancs kiadásával végezhető és az alábbi képet kapjuk.

–>fplot3d1(x,y,fff)



Végezetül egy másik függvényt is ábrázolunk, amelyre majd az optimalizálás tárgyalásánál vissza fogunk térni. Legyen az ábrázolandó függvény az alábbi

$$z = x^3 + y^3 - 27x - 3y$$

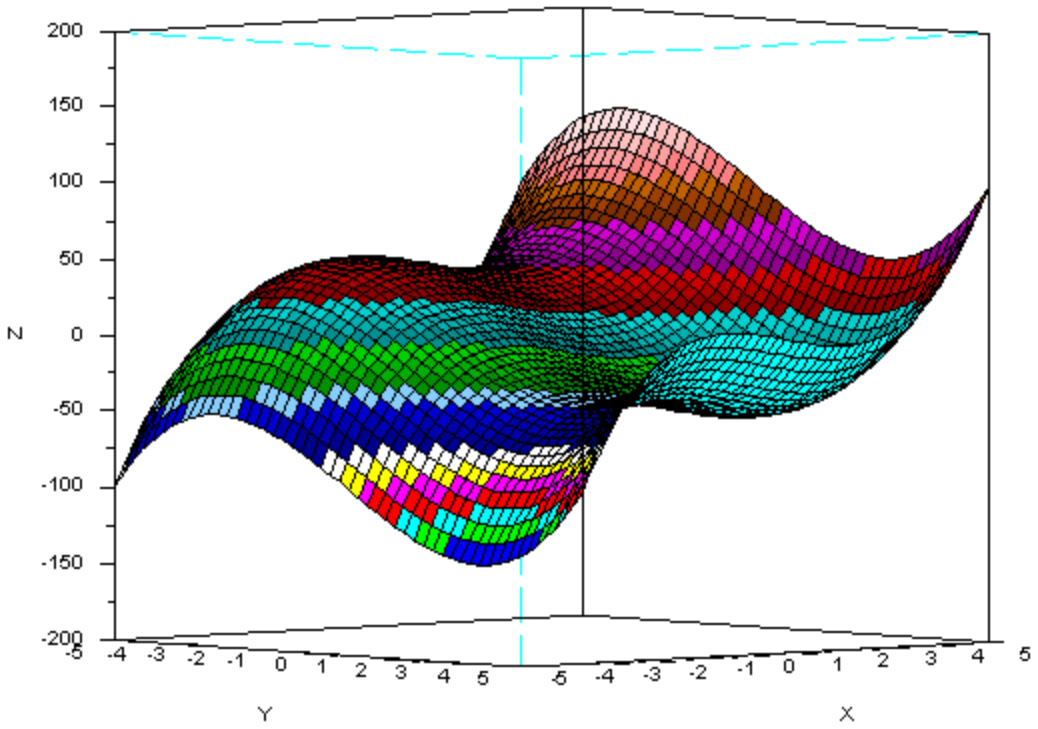
A szükséges lépések a következők:

–>deff('z=fv(x)','z=x^3+y^3-27*x-3*y')

–>x=[-5:0.1:5];
–>y=x;

–>fplot3d1(x,y,fv)

A függvény képe az alábbi ábrán látható. Megjegyezzük, hogy ennek a fügvénynek a (3,1) pontban minimuma, a (-3,-1) pontban maximuma, a (-3,1) és a (3,-1) pontokban nyeregpontja van, amelyek az ábrán is elég jól láthatók. A függvényértékek a fenti pontokban rendre -56, 56, 52, -52. Végül megjegyezzük, hogy nem törekedtünk a lehető legszebb megjelenítésre, mivel nem ez volt a célünk. Aki komolyan akar foglalkozni a Scilab programcsomag adta grafikus lehetőségekkel, szépíthet az ábrákon.


200
150
100
50
0
-50
-100
-150
-200
Z
-5 -4 -3 -2 -1 0 1 2 3 4 5
Y
-5 -4 -3 -2 -1 0 1 2 3 4 5
X