VÝUČBA DIFFERENCIÁLNEHO POČTU FUNKCIE VIAC PREMENNÝCH POMOCOU PG. SYST. MATHEMATICA

Monika Kováčová

Katedra Matematiky SjF STU Bratislava
kovacova_v@dekan.sjf.stuba.sk

Abstrakt. *V článku popisujeme možnosti použitia programového systému Mathematica pri výuke diferenciálneho počtu funkcie viac premenných. Ukážeme niekoľko nezávislých prístupov ku problému hľadania extrémov funkcie.*

Vývoj výpočtovej techniky je veľmi rýchly a výrazne ovplyvňuje spôsob výučby matematiky na VŠ [1,3,6]. Diferenciálny počet funkcií viac premenných je zaradený v sylaboch každého úvodného kurzu vysokoškolskej matematiky. Každý pedagóg z vlastnej skúsenosti pozná miesta pre študentov najproblematickejšie. Jedným z nich je určite otázka, ako derivovať funkcie viac premenných, druhým je hľadanie extrémov funkcie.

Hľadanie parciálnych derivácií sa dá naučiť. Potrebný je jedine tréning. Horšie je to s hľadaním extrémov. Existujú kroky výpočtu, ktoré permanentne už generáciám študentov spôsobujú problémy. Ak nejde o veľmi jednoduchý príklad, prvé parciálne derivácie sú spravidla komplikovanejšie a úloha vyriešiť systém $f'_x(x,y) = 0$, $f'_y(x,y) = 0$ býva problémom. Ďalším problematickým miestom býva výpočet hessiánu. Tu sú hlavnou príčinou numerické chyby.

K úlohe nájsť najväčšiu a najmenšiu hodnotu funkcie vedú mnohé praktické úlohy v oblasti techniky. V tomto článku by som chcela popísať dva možné prístupy hľadania extrémov pomocou programového systému Mathematica [5].

Použiť predefinované príkazy:

Mathematica ponúka veľké množstvo štandartne predefinovaných príkazov (má vytvorený kompletný metajazyk). Príkladom sú napr. ConstrainedMin, ConstrainedMax na hľadanie viazaných extrémov (riešenie úloh lineárneho programovania), vo verzii 3.1 aj príkazy na hľadanie minora matice.

Nie všetky témy sú však takto prepracované a je potrebné, aby pedagóg dokázal sám vytvoriť pre študentov podporný programový balík (package). Ukážku, ako ho vytvoriť uvádzame v 3. časti článku. Príprava výučby je v tomto prípade natoľko pohodlná, že skúsenému pedagógovi (s minimálnou programovou zručnosťou) nezaberie veľa času. Tento postup naviac umožňuje predkladať a riešiť aj ťažšie problémy. Výrazným spôsobom posúva hranicu, ktorú zvažuje pedagóg pri zaradení príkladu: význam $\leftrightarrow$ časová náročnosť.

Simulovať priebeh výpočtu:

Keď hľadáme lokálne extrémy funkcie jednotlivé kroky, ktoré by sme vykonávali pri ručnom počítaní uskutočníme pomocou programového systému Mathematica. Najskôr

definujeme funkciu, nájdeme prvé parciálne derivácie, zostavíme zodpovedajúci systém rovníc, vyriešime ho. Získame tak stacionárne body funkcie. Potom zostavíme hessián a určením jeho charakteru rozhodneme, v ktorých bodoch má funkcia maximá či minimá.

Tento postup je veľmi výhodný, najmä na cvičeniach, upevňuje správne techniky počítania a umožňuje precvičiť myšlienku výpočtu. Ukážku tohoto postupu nájdete v 4. časti tejto práce. Bolo by však nesmierne únavné, keby pedagóg pri príprave na cvičenie musel pri hľadaní vhodných úloh tento postup neustále opakovať. Vtedy je vhodnejšie použiť predprogramované balíky.

2.Formulácia matematického problému:

Nech funkcia *f* je 2x diferencovateľná funkcia n premenných. Funkcia *f* môže mať lokálny extrém len v tých bodoch, v ktorých sa všetky jej parciálne derivácie, ktoré existujú, rovnajú nule, alebo v tých bodoch, v ktorých neexistuje niektorá parciálna derivácia.

Bod, v ktorom má funkcia *f* parciálne derivácie podľa každej premennej a všetky tieto derivácie sa rovnajú nule, sa nazýva *stacionárny bod* funkcie *f*. Z definície stac. bodu vyplýva, že sa v ňom gradient funkcie, ak existuje rovná nule. V prípade funkcie dvoch premenných to geometricky znamená, že v bode *[x,y,f(x,y)]* má graf funkcie dotykovú rovinu, ktorá je rovnobežná so súradnicovou rovinou xy. Niektoré z týchto bodov sú bodmi lokálneho maxima, lok. minima resp. sedlové body. Stacionárnosť bodu A je teda nutnou podmienkou na to, aby funkcia diferencovateľná v bode A mala v tomto bode lokálny extrém. Táto podmienka však nie je postačujúca. Charakter bodov určuje hodnota hessiánu v stacionárnych bodoch. Hessián je štvorcová matica druhých parciálnych derivácií funkcie *f* vzhľadom na všetky premenné.

$$hessian(f) = \left| \frac{\partial^2 f}{\partial x_i \partial x_j} \right|$$

Analýza kritických bodov záleží na hodnotách minorov tejto matice v stacionárnych bodoch. Minory matice sú determinanty štvorcových submatíc tvorených postupne po diagonále z ľavého horného rohu.

Maticu nazývame *pozitívne definitnou* ak sú všetky minory kladné. Ak je hessián *f* v stacionárnom bode pozitívne definitnou maticou, potom v tomto stacionárnom bode funkcia *f* nadobúda *lokálne minimum*.

Maticu nazývame *negatívne definitnou*, ak jej minory presne striedajú znamienko, počnúc záporným znamienkom pre prvý subdeterminant (prvok $a_{1,1}$ hessovej matice). Ak hessián *f* v stacionárnom bode je negatívne definitná matica, potom v tomto bode funkcia *f* nadobúda lokálne maximum.

Ak hesián *f* v stacionárnom bode nie je ani pozitívne, ani negatívne definitná matica, potom tento stacionárny bod nazveme *sedlovým bodom* (saddle point). Nebudeme uvažovať prípad, keď hlavný minor je rovný 0. V prípade sedlového bodu, aby sme mohli popísať správanie funkcie v okolí tohoto bodu, musíme vypočítať vlastné hodnoty a vlastné vektory hessiánu v tomto bode. Ak je vlastná hodnota kladná (záporná) potom funkcia rastie (klesá) v smere príslušného vlastného vektora. Vlastnosť pozitívnej (negatívnej) definitnosti matice korešponduje s prípadom keď sú všetky vlastné hodnoty kladné (záporné).

3.Ukážka možnej prípravy podporného programového balíka

Ukážeme si ako možno vytvoriť pomocný programový balík, ktorý nájde stacionárne body funkcie *f* a vyšetrí ich charakter.

Definujme si patern funkcie. Musíme určiť, čo v našej definícii je definičný obor, aké je pravidlo priradenia, a aký je jej obor hodnôt.

In[1]:=

```
funkcia[old, rules, new];
```

In[2]:=

```
dom[fun_funkcia]:=fun[[1]]
rules[fun_funkcia]:=fun[[2]]
cod[fun_funkcia]:=fun[[3]]
```

Všetky nasledujúce príkazy budeme priebežne testovať na jednoduchom paraboloide, ktorý definujeme ako funkciu `fun1`

In[5]:=

```
fun1=funkcia[{x,y},{9-x^2-y^2},{z}]
```

Out[5]=

```
                          2    2
funkcia[{x, y}, {9 - x  - y }, {z}]
```

Definujme teraz gradient funkcie:

In[6]:=

```
grad[expr_,var_List]:=D[expr,#]& /@ var
```

In[7]:=

```
grad[rules[fun1],{x,y}]
```

Out[7]=

```
{{-2 x}, {-2 y}}
```

Ak by sme sa chceli vyhnúť takémuto komplikovanému spôsobu zadávania a rozhodneme sa prijať definíciu funkcie *In[1]* , stačí dodefinovať grad funkcie takto:

In[8]:=

```
grad[fun_funkcia]:=grad[rules[fun],dom[fun]]
grad[fun1]
```

Out[9]=

```
{{-2 x}, {-2 y}}
```

Všeobecná funkcia sa v pg. systéme Mathematica spravidla nazýva generická funkcia. Definujme si ju: (ak chceme pracovať s funkciou troch alebo viac premenných , nie je problém s úpravou definície, zvládne ju istotne každý)

In[10]:=

```
genericFun:=funkcia[{x,y},{f[x,y]},{z}]
```

Gradient takejto funkcie vypočítame:

In[11]:=

```
grad[genericFun]
```

Out[11]=

```
{{f(1,0)[x, y]}, {f(0,1)[x, y]}}
```

Tak ako sme už uviedli, kritické body sú body, v ktorých je gradient funkcie rovný nule. Vytvoríme teraz funkciu, ktorá takéto kritické body vyhľadá.

In[12]:=

```
criticalPoints[fun_funkcia]:=
    criticalPoints[fun]=Select[Union[
                    Solve[grad[fun]==0,dom[fun],
                     VerifySolutions->True]],FreeQ[#,Complex]&]
criticalPoints[fun1]
```

Out[13]=

```
{{x -> 0, y -> 0}}
```

Teraz nasleduje ťažšia časť úlohy. Je potrebné nájdené kritické body analyzovať a určiť ich charakter. V časti kde sme formulovali náš matematický problém sme uviedli postup, ktorým vieme vyšetriť charakter kritických bodov. Potrebujeme definovať hessián funkcie *f*. Rovnako ako v definícii gradientu, buď použijeme len základnú definíciu, alebo si situáciu opäť zjednodušíme tak, aby sme nemuseli zadávať jednotlivé premenné.

In[14]:=

```
hessian[fun_,var_]:=Outer[D[fun,#1,#2]&,var,var]
hessian[fun_funkcia]:=hessian[First[rules[fun]],dom[fun]]
hessian[fun1]
```

Out[16]=

```
{{-2, 0}, {0, -2}}
```

In[17]:=

```
hessian[genericFun]//TableForm
```

Out[17]//TableForm=

```
(2,0)          (1,1)
f      [x, y]  f      [x, y]

(1,1)          (0,2)
f      [x, y]  f      [x, y]
```

Nasledujúcim krokom bude nájdenie jednotlivých minorov matice. Najskôr nájdeme prvý minor matice, potom rekurentným postupom ostatné.

In[18]:=

```
oneMinor[matrix_]:= Map[Drop[#,-1]&,Drop[matrix,-1]]
allMinor[matrix_]:=Det /@  NestList[oneMinor,matrix,Length[matrix]-1]
allMinor[%16]
```

Out[19]=

```
{4, -2}
```

In[20]:=

```
allMinor[hessian[genericFun]]
```

Out[20]=

```
     (1,1)       2     (0,2)          (2,0)             (2,0)
{-f       [x, y]  + f       [x, y] f       [x, y],   f       [x, y]}
```

Testujme pozitívnu definitnosť hesiánu:

In[21]:=

```
pozitivneDefinitnaQ[matrix_]:=And @@    Positive[allMinor[matrix]]
```

In[22]:=

```
negativneDefinitnaQ[matrix_]:=pozitivneDefinitnaQ[-matrix]
```

Nájdeme všetky minimá a maximá funkcie pomocou nutnej a postačujúcej podmienky uvedenej pri formulácii problému.

In[23]:=

```
Minimum[fun_funkcia]:=Select[criticalPoints[fun],
    pozitivneDefinitnaQ[hessian[fun]/.#]&]
Maximum[fun_funkcia]:=Select[criticalPoints[fun],
    negativneDefinitnaQ[hessian[fun]/.#]&]
```

Funkčnosť definovaných príkazov overíme na našej funkcii fun1

In[25,26]:=

```
Minimum[fun1]
Maximum[fun1]
```

Out[25,26]=

```
{}
{{x -> 0, y -> 0}}
```

Ak hessián v stacionárnom bode nie je ani pozitívne, ani negatívne definitná matica, nazvali sme tento bod sedlovým. Nasleduje procedúra, ktorá vyšetruje charakter týchto bodov.

Najskôr z množiny všetkých stacionárnych bodov vyberieme tie, ktoré nie sú ani bodmi minima, ani maxima.

In[27]:=

```
ostatnebody[fun_funkcia]:=Complement[criticalPoints[fun],
    Minimum[fun],Maximum[fun]];
```

Budeme hľadať vlastné hodnoty a vlastné vektory príslušného hessiánu.

In[28]:=

```
saddlePoints[fun_funkcia]:=
    With[{ostatne=ostatnebody[fun]},
        If[ostatne=={},{},
           Thread[Sedla[ostatne,
           Transpose[Eigensystem[#]]&/@(hessian[fun]/.ostatne)]]]];
```

pozn.: Ak charakter riešených príkladov vyžaduje numerické riešenie, zmeníme jednotlivé príkazy na NcriticalPoints...

Programový balík máme vytvorený. Jeho globálne použitie si ukážeme na ďalšom príklade.

In[29]:=

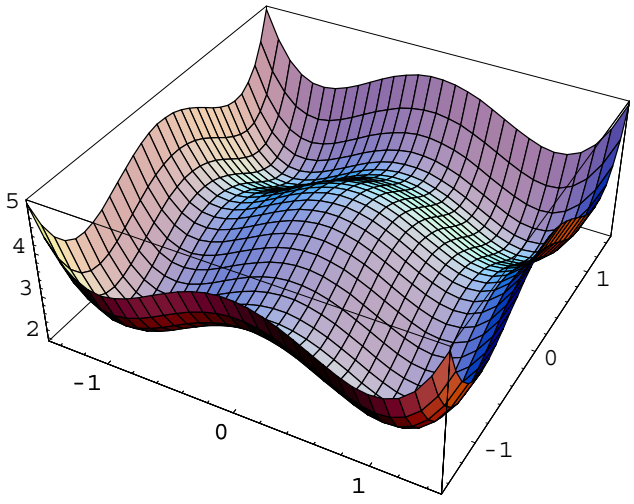
```
fun2=funkcia[{x,y},{x^4+y^4-2x^2-2y^2+4},{z}]
```

Out[29]=

```
                          2    4       2    4
funkcia[{x, y}, {4 - 2 x  + x  - 2 y  + y }, {z}]
```

In[30]:=

```
Plot3D[x^4+y^4-2x^2-2y^2+4,{x,-1.5,1.5},{y,-1.5,1.5},
        PlotPoints->30]
```



Out[30]=

```
-SurfaceGraphics-
```

In[31]:=

```
Minimum[fun2]
```

Out[31]=

```
{{x -> -1, y -> -1}, {x -> -1, y -> 1},

 {x -> 1, y -> -1}, {x -> 1, y -> 1}}
```

In[32]:=

```
Maximum[fun2]
```

Out[32]=

```
{{x -> 0, y -> 0}}
```

In[33]:=

```
saddlePoints[fun2]
```

Out[33]=

```
{Sedla[{x -> -1, y -> 0}, {{-4, {0, 1}}, {8, {1, 0}}}],
 Sedla[{x -> 0, y -> -1}, {{-4, {1, 0}}, {8, {0, 1}}}],
 Sedla[{x -> 0, y -> 1}, {{-4, {1, 0}}, {8, {0, 1}}}],
 Sedla[{x -> 1, y -> 0}, {{-4, {0, 1}}, {8, {1, 0}}}]}
```

Jedine sedlové body sú možno komplikovanejšie čitateľné. Zistili sme sme, že jeden zo sedlových bodov má súradnice [-1,0] a funkcia v smere osi y-ovej klesá a v smere osi x-ovej rastie. Podobne ľahko prečítame výsledok aj pre ostatné sedlové body.

4.Priamy spôsob výpočtu

Kroky, ktoré budeme postupne vykonávať sú ľahšie pochopiteľné z programátorského hladiska. Rovnako sú potrebné podstatne menšie vedomosti z programového systému Mathematica, aby sme mohli používať tento postup.

Definujme si funkciu (štandartný postup)

In[34]:=

```
f[x_,y_]:= x^4+y^4-2x^2-2y^2+4
```

Vypočítame prvé parciálne derivácie tejto funkcie

In[35,36]:=

```
D[f[x,y],x]
D[f[x,y],y]
```

Out[35,36]=

```
          3
-4 x + 4 x
          3
-4 y + 4 y
```

Zostavíme a vyriešime systém rovníc *grad f = 0*

In[37]:=

```
Solve[{-4x+4x^3==0,-4y+4y^3==0},{x,y}]
```

Out[37]=

```
{{x -> -1, y -> -1}, {x -> -1, y -> 0}, {x -> -1, y -> 1},
 {x -> 0, y -> -1}, {x -> 0, y -> 0}, {x -> 0, y -> 1},
 {x -> 1, y -> -1}, {x -> 1, y -> 0}, {x -> 1, y -> 1}}
```

Spočítame hessián funkcie *f* a dosadením určíme charakter stacionárnych bodov.

In[38]:=

```
Det[{{D[f[x,y],x,x], D[f[x,y],x,y]},{ D[f[x,y],y,x], D[f[x,y],y,y]}}]
```

Out[38]=

```
          2        2         2  2
16 - 48 x  - 48 y  + 144 x  y
```

In[39,40]:=

```
%65/.%64
D[f[x,y],x,x]/.%64
```

Out[39,40]=

```
{64, -32, 64, -32, 16, -32, 64, -32, 64}
{8, 8, 8, -4, -4, -4, 8, 8, 8}
```

Získaný výsledok stačí len správne interpretovať. Začneme čítať : bod [-1,-1] je bodom minima, bod [-1,0] je sedlový bod ...

5. Záver.

K otázke, čo z matematiky absolventi technického štúdia skutočne využijú v praxi sa prirodzeným spôsobom pridružuje ďalšia, akým spôsobom ich to naučiť. V súčasnej dobe nie je použitie PC na teoretických cvičeniach ešte slále samozrejmosťou. Je to najmä preto, lebo ešte stále je nedoriešená otázka, akým spôsobom a do akej miery používať programové produkty, ktoré študentovi aj bez dostatočnej znalosti algoritmu výpočtu poskytnú výsledky riešenia úlohy [1,2,4,6].

Programový systém Mathematica nenahrádza výuku matematiky na VŠ, ale pri jej výuke pomáha. Je pomôckou, ktorá odstaňuje mechanické počítanie a umožňuje väčšiu možnosť výberu príkladov, v ktorých je dôležité najmä logické uvažovanie, technickú stránku pomože zrealizovať Mathematica. Ďalšou nesmiernou výhodou je skutočnosť, že Mathematica nie je výukový software. Jej použitie nie je obmedzené len na školské úlohy a preto ak poskytneme študentom možnosť pracovať s takýmto produktom, zlepší sa ich pripravenosť riešiť úlohy praxe.

Literatúra

1. Halada, L.: Stabilita úloh a algoritmov vo výučbe numerickej matematiky na SjF STU, Informatika a Algoritmy, Prešov, 3. -4. September 1998.
2. Halada, L.; Kováčová, M.: Skúsenosti s použitím programového systému Mathematica pri výučbe numerickej matematiky na Sjf STU, Matematická štatistika a Numerická Matematika, Kálnica, 1. - 5. Júna 1998.
3. Kolesárová, A ., Kováčová, M., Záhonová, V.: Použitie programového systému Mathematica pri výučbe základov matematickej analýzy na SjF STU, Matematická štatistika a Numerická matematika, Kalnica, 1.-5. Júna 1998.
4. Kováčová, M.: Prípravné materiály experimentálnej výučby na SjF STU.
5. Wolfram Research: The Mathematica Book 3rd ed.,Wolfram Media/Cambridge University Press, 1996.
6. Záhonová, V.: Výučba integrálneho počtu funkcie jednej reálnej premennej s podporou programového systému Mathematica, In.:25 VŠTEP-Z Matematika v inžinierskom vzdelávaní, Trnava, 7. - 10. September 1998.