Metóda najmenších štvorcov

Príklad 1

```
» x=1:10; y=rand(1,10)*20-10;
```

Zadané hodnoty x,y budeme postupne aproximovať polynómami rôznych stupňov. Najjednoduchšie – z hľadiska náročnosti príkazov v Matlabe – je postupovať analogickou cestou ako bol výpočet interpolačného polynómu cez Van der Mondovu maticu.

```
» V=vander(x);
```

Táto matica nám dovoľuje nájsť interpolačný polynóm 9 stupňa. Ak chceme aproximovať dané hodnoty polynómom nižšieho stupňa, musíme na to použiť príslušný menší počet stĺpcov matice V.

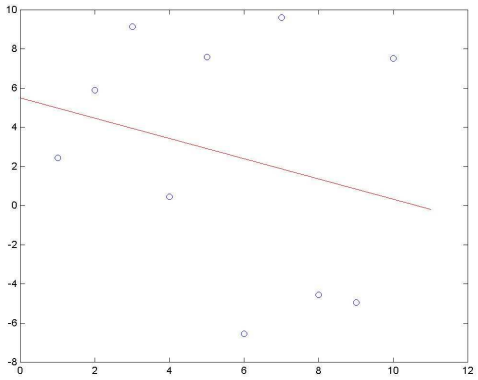
Lineárna aproximácia – použijeme posledné dva stĺpce Van der Mondovej matice V (matlab vyrába Van der Mondovu maticu tak, že najvyššie mocniny x sú vľavo). Možnosti sú dve, ide totiž o hľadanie zovšeobecneného riešenia:

```
» a=V(: , 9:10)\y'
                    a =
                      -5.176064950438405e -001
                       5.502693241403512e+000

» aa=pinv(V(:,9:10))*y'
                        aa =
                          -5.176064950438405e-001
                           5.502693241403510e+000
```

Reziduálny súčet štvorcov:

```
» rs=polyval(a,x)-y; rss=rs*rs'

rss =    3.262546301265631e+002
```

Obrázok:



Obrázok ukazuje, že skúšať to s priamkou je dosť zúfalé…
Skúsme 3. stupeň:

```
» a=V(:,7:10)\y'
» rs=polyval(a,x)-y; rs*rs'
                                    ans =    2.599959839750684e+002
```

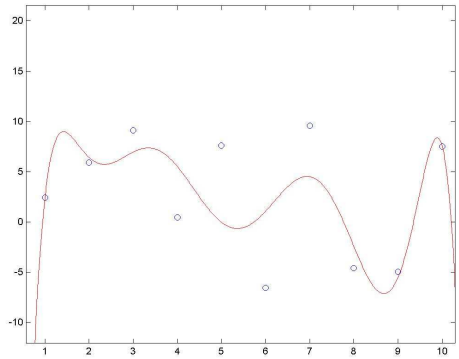
RSS je stále veľký, to ani nejdeme kresliť…

Pridajme – 7. stupeň:

```
» a=V(:,3:10)\y'
» rs=polyval(a,x)-y; rs*rs'
                                    ans =    1.882033514323069e+002
```

Stále je to veľa…
Už sme na konci – ešte 8. stupeň a koniec. 9. stupeň je interpolácia...

```
» a=V(:,2:10)\y'
» rs=polyval(a,x)-y; rs*rs'
                        ans =    1.778851326905628e+002
```



RSS je v druhom prípade o kúsok menšie, ale nestojí to za reč. Sú to katastrofálne hodnoty, zadanie vyzerá ako mnš-odolné. S rastúcim stupňom polynómu sa RSS zmenšuje len málo. Samozrejme, 9. stupeň to okamžite zrazí na nulu (vyskúšajte).

Odolnosť zadaných hodnôt voči pokusom ich aproximovať je pochopiteľná. Získali sme ich cez rand, a ak sú hodnoty skutočne náhodné, znamená to práve to, že sa vymykajú akejkoľvek pravidelnosti vyjadriteľnej funkčným predpisom.

Príklad 2:

Trochu lepšie zadanie získame, ak napr. usporiadame hodnoty y podľa veľkosti.

```
» y=sort(y)
```

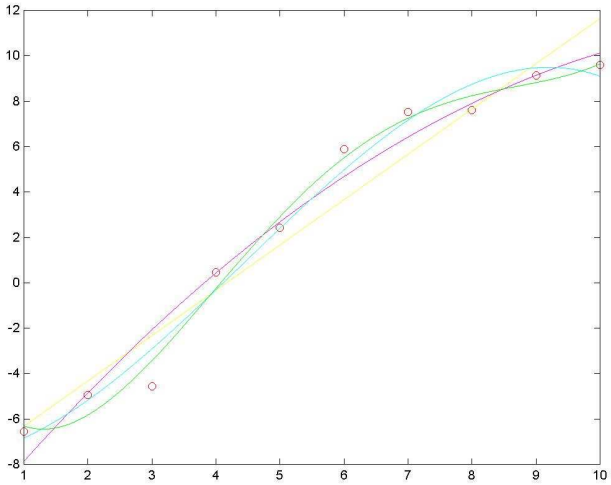
Pozrime sa jedným šupom na rss pri aproximácii polynómami rôznych stupňov. Začneme priamkou a skončíme 8. stupňom:

```
» for i=9:-1:2, a=V(:,i:10)\y'; rs=polyval(a,x)-y; rss=rs*rs'; [10-i, rss], end
```

stupeň polynómu	RSS
1.000000000000000e+000	1.959035262430398e+001
2.000000000000000e+000	1.107274462344985e+001
3.000000000000000e+000	6.115159652853779e+000
4.000000000000000e+000	3.582761215320212e+000
5.000000000000000e+000	3.448341620149558e+000
6.000000000000000e+000	2.477046211370157e+000
7.000000000000000e+000	2.425027888889459e+000
8.000000000000000e+000	6.051998040498070e -001

Tieto hodnoty RSS sú už prijateľnejšie. Po 4. stupeň to klesá vždy zhruba na polovicu, potom sa tempo klesania spomalí. Až pri 8. stupni to výraznejšie poskočí k nule, ale to už je takmer interpolácia. Nakreslíme si prvé štyri aproximácie:

```
» xh=1:0.001:10; plot(x,y,'or'), hold on
» a=V(:,9:10)\y'; yh=polyval(a,xh); plot(xh,yh,'y')
» a=V(:,8:10)\y'; yh=polyval(a,xh); plot(xh,yh,'m')
» a=V(:,7:10)\y'; yh=polyval(a,xh); plot(xh,yh,'c')
» a=V(:,6:10)\y'; yh=polyval(a,xh); plot(xh,yh,'g')
```



Príklad 3

Vráťme sa k tomu príkladu, ktorý robil pri interpolácii problémy s presnosťou.

```
» x=1:20; y=exp(0.1*x)+rand(1,20);
» V=vander(x);
```

Preskúmame RSS pri aproximácii polynómami 1. až 18. stupňa. Bude to síce dlhý výpis, ale stojí za to ho preštudovať:

```
» for i=19:-1:2, a=V(:,i:20)\y'; rs=polyval(a,x)-y; rss=rs*rs'; [20-i,rss],  end

ans =    1.000000000000000e+000    6.274955927767788e+000
ans =    2.000000000000000e+000    1.644107084126936e+000
ans =    3.000000000000000e+000    1.558371905607419e+000
ans =    4.000000000000000e+000    1.047118557030246e+000
ans =    5.000000000000000e+000    1.046545412058349e+000
ans =    6.000000000000000e+000    1.032600635288216e+000
ans =    7.000000000000000e+000    9.971772220130576e-001
ans =    8.000000000000000e+000    9.414805580096204e-001
ans =    9.000000000000000e+000    9.233544399124061e-001
Warning: Rank deficient, rank = 10  tol =   5.6333e-002.
ans =    1.000000000000000e+001    9.087635163598656e-001
Warning: Rank deficient, rank = 9  tol =   1.0988e+000.
ans =    1.100000000000000e+001    1.559309193626851e+000
Warning: Rank deficient, rank = 8  tol =   2.1507e+001.
ans =    1.200000000000000e+001    6.901008178003219e+000
Warning: Rank deficient, rank = 8  tol =   4.2214e+002.
ans =    1.300000000000000e+001    9.604328892707645e+000
Warning: Rank deficient, rank = 7  tol =   8.3055e+003.
ans =    1.400000000000000e+001    2.057482736968469e+001
Warning: Rank deficient, rank = 7  tol =   1.6374e+005.
ans =    1.500000000000000e+001    2.402543507455322e+001
Warning: Rank deficient, rank = 7  tol =   3.2336e+006.
ans =    1.600000000000000e+001    2.721280115844329e+001
Warning: Rank deficient, rank = 7  tol =   6.3954e+007.
ans =    1.700000000000000e+001    3.016957651875442e+001
Warning: Rank deficient, rank = 7  tol =   1.2665e+009.
ans =    1.800000000000000e+001    3.294777511619407e+001
```

Vidíme, že polynóm 2. stupňa predstavuje hmatateľný pokrok voči aproximácii priamkou. Ďalej však, najmä od 5. stupňa, už RSS klesá veľmi neochotne, skoro až nijak. To neprekvapuje, daná funkcia v sebe skrýva exponenciálu a tá sa veľmi neráči s polynómami. Zaujímavejší je fakt, že od 9. stupňa matlab začne hlásiť nedostatky v regularite matice, a že to nemyslí ako planú hrozbu vidno z toho, že RSS začína rásť.

Skúsime teda šťastie s iným prístupom – cez systém normálnych rovníc. Najprv zistíme, ako bude vyzerať aproximácia pre 1. a 2. stupeň polynómu.

1. stupeň. Vyrobíme normálny systém Ax=b a budeme ho riešiť:

```
» A(1,1)=length(x); sx1=sum(x); A(1,2)=sx1; A(2,1)=sx1; A(2,2)=x*x';
» b(1,1)=sum(y); b(2,1)=x*y';
» a=A\b
```

```
a =       4.528691792517926e-001
          3.273043596417055e-001
```

Maticu sme zostavili tak, ako sme zvyknutí z teoretických cvičení. Koeficienty a preto vychádzajú v poradí od absolútneho člena ku koeficientu pri najvyššej mocnine. Matlabovské vyčísľovanie polynómov však pracuje s opačným poradím koeficientom. Na ľavopravé prehodenie poradia čísel v matici slúži príkaz *fliplr*, na horedolné slúži *flipud*.

```
» a=flipud(a); rs=polyval(a,x)-y; rss=rs*rs'

rss =    6.274955927767786e+000
```

Vidíme, že RSS je až na posledné miesto výpisu rovnaký ako pri výpočte predošlou metódou. Kreslíme:

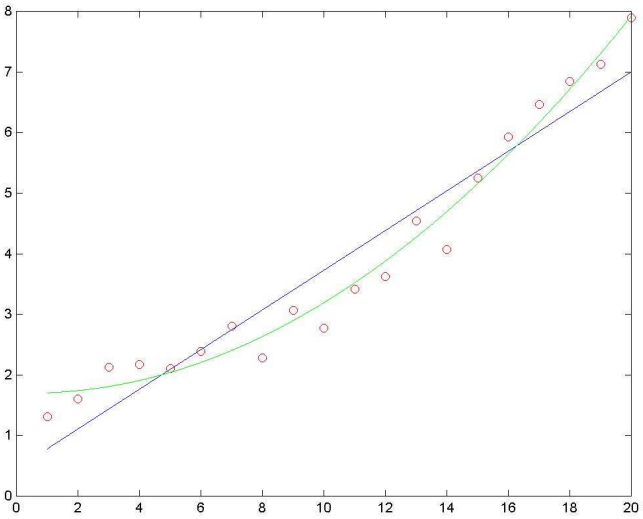
```
» xh=1:0.01:20; yh=polyval(a,xh); plot(x,y,'or', xh,yh)
```

2. stupeň.

```
» A(3,1)=A(2,2); A(1,3)=A(2,2);
sx3=(x.*x)*x'; A(3,2)=sx3; A(2,3)=sx3; A(3,3)=(x.*x)*(x.*x)';
» b(3,1)=(x.*x)*y';
» a=A\b
          a =       1.703439857684603e+000
                   -1.376037083997021e-002
                    1.624117764198456e-002

» a=flipud(a); rs=polyval(a,x)-y; rss=rs*rs'

rss =    1.644107084126935e+000
```

Kreslíme:

```
» hold on, yh=polyval(a,xh); plot(xh,yh,'g')
```



Teraz nás zaujíma, aké RSS budú dávať vyššie stupne aproximačného polynómu. Samozrejme, počítať to stupeň za stupňom nie sme ochotní.

Hneď na začiatok vyrobíme maticu A a pravú stranu b pre 18. stupeň. Z týchto hodnôt si budeme potom krok za krokom vyberať toľko, koľko bude treba. V matici A máme súčty mocnín x, najvyššou je suma $x^{18}*x^{18}$. Vyrobíme maticu A (na papieri si skúste, čo nasledujúce príkazy robia):

```
» rx=ones(1,20); XX(1,:)=rx; for i=2:19, rx=rx.*x; XX(i,:)=rx; end
» A=XX*XX';
```

Podobne vznikne stĺpec b:

```
» b=XX*y';
```

Vypočítame teraz jedným cyklom RSS pre všetky aproximačné polynómy stupňa 1 až 18.

```
» for i=2:19, a=A(1:i,1:i)\b(1:i,1); a=flipud(a); rs=polyval(a,x)-y; rss=rs*rs'; [i-1,rss], end

ans =   1.000000000000000e+000   6.274955927767790e+000
ans =   2.000000000000000e+000   1.644107084126934e+000
ans =   3.000000000000000e+000   1.558371905607418e+000
ans =   4.000000000000000e+000   1.047118557030250e+000
ans =   5.000000000000000e+000   1.046545412058347e+000
ans =   6.000000000000000e+000   1.032600635288197e+000
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 6.454341e-018.
ans =   7.000000000000000e+000   9.971772220130527e-001
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 1.751308e-020.
ans =   8.000000000000000e+000   9.414805580173554e-001
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 4.689840e-023.
ans =   9.000000000000000e+000   9.233544414821504e-001
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 1.242940e-025.
ans =   1.000000000000000e+001   8.990694745669929e-001
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 3.266769e-028.
ans =   1.100000000000000e+001   1.474355733313504e+000
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 3.815983e-036.
ans =   1.200000000000000e+001   7.617919027254679e-001
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 2.673051e-039.
ans =   1.300000000000000e+001   7.628821929204142e-001
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 4.097714e-042.
ans =   1.400000000000000e+001   7.599192195494768e-001
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 5.006085e-044.
ans =   1.500000000000000e+001   7.580655865797055e-001
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 1.064238e-046.
ans =   1.600000000000000e+001   7.493070065513141e-001
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 2.829644e-049.
ans =   1.700000000000000e+001   7.390903725273923e-001
        Warning: Matrix is close to singular or badly scaled.
                 Results may be inaccurate. RCOND = 4.072813e-052.
ans =   1.800000000000000e+001   7.457640908819694e-001
```

Ten výpis vyzerá desivo. Už od 7. stupňa sa matlab sťažuje na zlé podmienky práce. Hodnoty RSS sú samozrejme lepšie ako pri výpočte cez neúplnú VanDerMondovu maticu, ale takisto v 11. kroku začínajú stúpať, čo je jednoznačný znak toho, že každým vyšším stupňom polynómu miera nepresnosti presahuje mieru priblíženia sa k daným bodom.

Na ručný výpočet je systém normálnych rovníc pohodlnejší ako riešenie preurčenej sústavy. Trpí však ešte viac než predošlý postup prítomnosťou malých a zároveň relatívne veľmi veľkých hodnôt v matici sústavy. To znemožňuje v prípade veľkého počtu hodnôt x a ich veľkého rozpätia počítať aproximácie polynómami vyšších stupňov.

Praktické riešenie týchto nedostatkov je pomerne jednoduché. Vo väčšine reálnych problémov je zaujímavá a potrebná iba aproximácia polynómami nižších stupňov – výrazne nižších než je počet nameraných hodnôt. Pri týchto úlohách, ako možno vidieť na porovnaní RSS pre stupne 1 až 6, sú výsledky oboch metód porovnateľné.

Nakoniec vyskúšajme obe metódy na jednoduchšom príklade, kde nebudeme narážať na hranice presnosti.

Príklad 4

```
» x=rand(1,12)*5; x=sort(x);
» y=rand(1,12)+log(x)*2;
» for i=11:-1:2, a=V(:,i:12)\y'; rs=polyval(a,x)-y; rss=rs*rs'; [12-i,rss],  end

ans =   1.000000000000000e+000   7.098594580710850e+000
ans =   2.000000000000000e+000   2.512438441533657e+000
ans =   3.000000000000000e+000   5.022992925111937e-001
ans =   4.000000000000000e+000   4.753805313934577e-001
ans =   5.000000000000000e+000   4.750025613897964e-001
ans =   6.000000000000000e+000   2.537108995456767e-001
ans =   7.000000000000000e+000   2.262339451494990e-001
ans =   8.000000000000000e+000   1.647724888653913e-001
ans =   9.000000000000000e+000   8.006897927964332e-002
ans =   1.000000000000000e+001   2.131263731384051e-002
```

Výsledky sú prijateľné, RSS klesá. Skúsme normálny systém:

```
» clear XX rx
» rx=ones(1,12); XX(1,:)=rx; for i=2:11, rx=rx.*x; XX(i,:)=rx; end
» clear A b, A=XX*XX‘; b=XX*y';
» for i=2:11, a=A(1:i,1:i)\b(1:i,1); a=flipud(a); rs=polyval(a,x)-y; rss=rs*rs'; [i-1,rss], end

» for i=2:11, a=A(1:i,1:i)\b(1:i,1); a=flipud(a); rs=polyval(a,x)-y; rss=rs*rs'; [i-1,rss], end

ans =   1.000000000000000e+000   7.098594580710850e+000
ans =   2.000000000000000e+000   2.512438441533656e+000
ans =   3.000000000000000e+000   5.022992925111941e-001
ans =   4.000000000000000e+000   4.753805313934583e-001
ans =   5.000000000000000e+000   4.750025613897963e-001
ans =   6.000000000000000e+000   2.537108995456747e-001
ans =   7.000000000000000e+000   2.262339451494962e-001
ans =   8.000000000000000e+000   1.647724889111017e-001
ans =   9.000000000000000e+000   1.130151396954471e-001
ans =   1.000000000000000e+001   1.400577629194843e-001
```

Porovnajme hodnoty RSS získané obidvoma postupmi. Väčšiu hodnotu zvýraznýme farebne:

7.098594580710850e+000	7.098594580710850e+000
2.512438441533657e+000	2.512438441533656e+000
5.022992925111937e-001	5.022992925111941e-001
4.753805313934577e-001	4.753805313934583e-001
4.750025613897964e-001	4.750025613897963e-001
2.537108995456767e-001	2.537108995456747e-001
2.262339451494990e-001	2.262339451494962e-001
1.647724888653913e-001	1.647724889111017e-001
8.006897927964332e-002	1.130151396954471e-001
2.131263731384051e-002	1.400577629194843e-001

Pri nižších stupňoch aproximačného polynómu sú obidva prístupy porovnateľné alebo dokonca systém normálnych rovníc je o málo presnejší. Pri 8. stupni sa situácia mení a v 9., 10. stupni sa výsledky sústavy normálnych rovníc musia brať s riadnou rezervou.

Úlohy: Nakreslite grafy aproximačných polynómov stupňa 2 až 6 (rôznymi farbami).

Počítajte systém normálnych rovníc pre polynóm stupňa 11. Mal by teoreticky vyjsť interpolačný polynóm. Vyjde?

Nakreslite graf polynómu, ktorý ste dostali, položený medzi body x,y. Zhodnoťte to, čo vidíte. Pridajte graf interpolačného polynómu získaného cez Vandermondovu sústavu a porovnajte oba grafy.

Príklad 5

Metóda najmenších štvorcov sa neviaže principiálne na polynómy. Aproximovať dané hodnoty môžeme s použitím ľubovoľnej rozumne zvolenej množiny bázových funkcií. Uplatniť môžeme pritom obidva vyššie uvedené postupy, samozrejme s príslušnými úpravami.

Majme zadané body x a v nich namerané hodnoty y:

```
>> x=sort(rand(14,1)*10-5);
>> y=x.*exp(x)+x.^2+rand(14,1)*2;
```

Pokúsime sa nájsť čo najlepšiu aproximáciu v tvare

$$f=a1*f1+a2*f2+a3*f3+a4*f4$$

kde:

```
>> f1=inline('2.^x');
>> f2=inline('3.^x');
>> f3=inline('x.^3');
>> f4=inline('1');
```

A. Vandermondovu maticu v pravom zmysle, ako sme ju robili doteraz, urobiť nevieme a rovnako nevyužijeme príkaz vander. Musíme sa vrátiť k základnej formulácii úlohy – k preurčenej sústave rovníc:

$$f(x)=y$$

čo po rozpísaní dáva:

$$[f1(x) \quad f2(x) \quad f3(x) \quad f4(x)] * [a1\ a2\ a3\ a4]' = y$$

Riešenie sústavy v matlabe:

```
>> A=[f1(x)  f2(x)  f3(x)  ones(length(x),1)];
>> a=A\y
                a =
                     -3.401030268604144e+000
                      3.703555275763195e+000
                     -1.976289877974936e-001
                      1.868016742883224e+000

>> rs=A*a-y; rss=rs'*rs
                rss =       1.395422361732643e+001
```

B. To isté riešme cez sústavu normálnych rovníc:

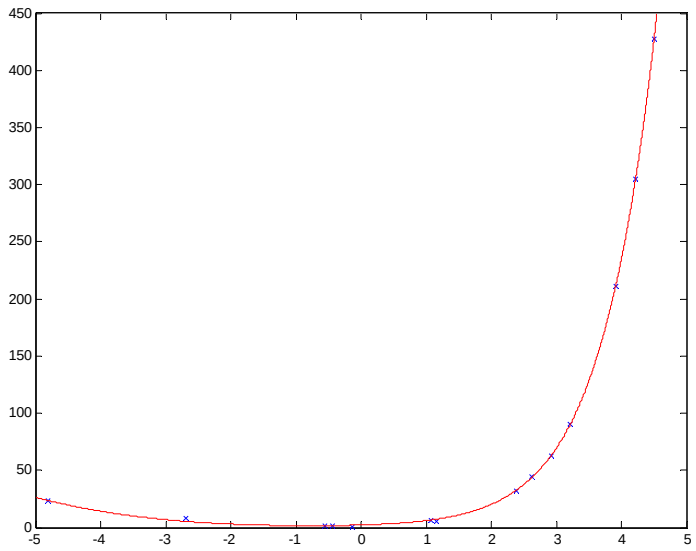
```
>> S=A'*A;
>> b=A'*y;
>> a=S\b
```

```
a =
   -3.401030268604387e+000
    3.703555275763232e+000
   -1.976289877974905e-001
    1.868016742883660e+000

>> rs=A*a-y; rss=rs'*rs
rss =    1.395422361732649e+001
```

Výsledky oboch metód sa líšia na posledných 2 až 3 miestach výpisu. RSS je menší v prvom prípade, ale ide o nebadateľný rozdiel na poslednej pozícii.

Obrázok:

```
>> xh=-5:0.001:5;
>> yh=[f1(xh')  f2(xh')  f3(xh')  ones(length(xh),1)]*a;
>> plot(x,y,'x'),hold on, plot(xh, yh', 'r')
```



Úlohy:

1. Nájdite aproximáciu daných hodnôt x,y s použitím bázových funkcií

```
>> f1=inline('2.^x');
>> f2=inline('4.^x');
>> f3=inline('1./x');
>> f4=inline('x.^4');
>> f5=inline('1+1./x.^2');
```

2. Aproximujte dané hodnoty x,y polynómom vhodného stupňa tak, aby ste dosiahli RSS porovnateľné alebo lepšie ako v príklade 5.