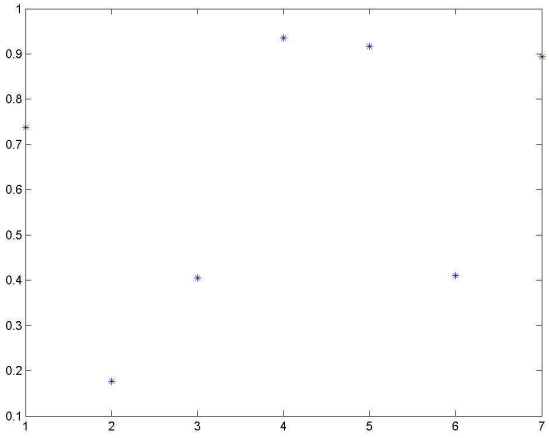
Interpolácia – VanDerMondova matica

1. Začnime jednoduchým príkladom, pri ktorom je možné sledovať, čo sa presne deje.

```
>> x=1:7;
>> y=rand(1,7)                    //vyrobí náhodný vektor dĺžky 7

      y =    0.7382    0.1763    0.4057    0.9355    0.9169    0.4103    0.8936
```

Obrázok:

```
>> plot(x,y,'*')          // parameter * spôsobí vykreslenie samotných bodov bez spájania, v tvare *
>> hold on                //tento príkaz zariadi, aby ďalšie príkazy "plot" kreslili do existujúceho obrázku
```



Chceme nájsť polynóm, ktorý v daných bodoch x bude mať uvedené hodnoty y.
Postup je jednoduchý:

```
>> V=vander(x)
                                V =
                                     1         1         1         1         1         1         1
                                    64        32        16         8         4         2         1
                                   729       243        81        27         9         3         1
                                  4096      1024       256        64        16         4         1
                                 15625      3125       625       125        25         5         1
                                 46656      7776      1296       216        36         6         1
                                117649     16807      2401       343        49         7         1
>> a=V\y'
                          a =
                            -1.052215999595353e-003
                             3.265075552934774e-002
                            -3.573508600932110e-001
                             1.737642132648633e+000
                            -3.718241690382854e+000
                             2.843668129254977e+000
                             2.008909948533674e-001
```

Vektor *a* obsahuje koeficienty hľadaného polynómu, pričom prvý koeficient je ten pri najvyššej mocnine. Hodnoty polynómu p(x) v rôznych bodoch prezradí funkcia polyval.

Pozrime sa na hodnotu polynómu v bode 4 – bude rovnaká ako pôvodné y(4)?

```
>> polyval(a,4)
                    ans =   9.354696991076125e-001
>> y(4)
                    ans =   9.354696991076055e-001
```

Zdá sa, že je to ono, hoci na posledných troch miestach sa to rozchádza.
Chceli by sme skontrolovať všetkých 7 bodov, ale nechce sa nám opakovať procedúru stále znovu. Matlab to našťastie dokáže jednou ranou:

```
>> [polyval(a,x); y]
                                ans =
 Columns 1 through 4

   7.382072458106651e-001   1.762661444946184e-001   4.057062130621020e-001   9.354696991076125e-001
   7.382072458106653e-001   1.762661444946180e-001   4.057062130620955e-001   9.354696991076055e-001

 Columns 5 through 7

   9.169044399134160e-001   4.102702069909062e-001   8.936495309134379e-001
   9.169044399134076e-001   4.102702069909454e-001   8.936495309135336e-001
```

V hornom riadku máme hodnoty polynómu v bodoch x, pod tým sú pôvodné hodnoty y. Sedí to, ale nie až tak presne, ako by sme snáď naivne očakávali. Aby sme boli konkrétni:

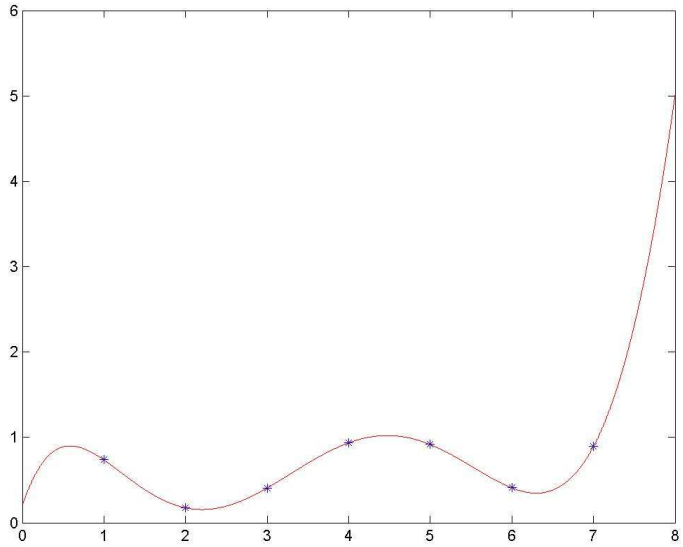
```
>> rozdiel=(polyval(a,x)- y)'
                                rozdiel =
                  -2.220446049250313e-016
                   4.440892098500626e-016
                   6.494804694057166e-015
                   6.994405055138486e-015
                   8.326672684688674e-015
                  -3.913536161803677e-014
                  -9.570122472268849e-014
```

Pri najnižších hodnotách je nepresnosť rovná „praktickej nule“, sú to hodnoty nie veľmi vzdialené od hraníc presnosti determinovanej množinou pc čísel, s akou pracuje ML. Väčší rozdiel je pri najvyšších hodnotách x. Chyba vzniká už pri výpočte koeficientov (riešení sústavy rovníc) a zvyšuje sa ešte pri dosadení do polynómu.

Úlohu ukončíme obrázkom – nakreslíme graf polynómu p(t) = a * [t^6, t^5, …, t, 1]’ .

Musíme si spomenúť, ako ML kreslí. Aby sme dostali pekný graf polynómu, rozhodne nevystačíme s pôvodným vektorom x, ale budeme potrebovať hustejšie delenie. Zároveň si neodpustíme zvedavý pohľad trochu viac vľavo a vpravo od hodnôt 1 a 7, čiže tzv. extrapoláciu:

```
>> xh=0:0.001:8;
>> yh=polyval(a,xh);
>> plot(xh, yh, 'r'), shg
```



2 Druhý príklad budeme zaznamenávať stručnejšie:

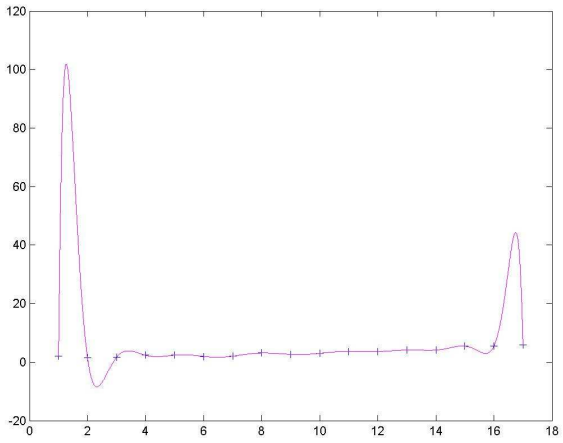
```
>> x=1:17;
>> y=exp(0.1*x)+rand(1,17);
>> V=vander(x);
>> a=V\y';
        Warning: Matrix is close to singular or badly scaled. Results may be inaccurate. RCOND = 3.307882e-
        026. (Type "warning off MATLAB:nearlySingularMatrix" to suppress this warning.)
>> z=polyval(a,x);
```

Dostali sme sa ku kritickému momentu VanDerMondovej metódy. Pre veľké matice V je riešenie sústavy s neznámymi koeficientami polynómu poznamenané dosť veľkou chybou. Matlab nám stihol povedať svoje, pozrime sa, čo to konkrétne znamená:

```
>> (z-y)'
                ans =    2.106848029370667e-011
                         5.795364188543317e-013
                         7.942273505534558e-010
                        -1.266571292291019e-009
                         7.862728246266215e-009
                         1.416762929196125e-008
                         9.084572294426607e-008
                        -2.727908761102071e-007
                        -3.112151167883326e-007
                        -2.582288129993060e-007
                         3.666492213394434e-006
                         7.486037789661282e-008
                        -8.376278370469947e-006
                         5.248902423815594e-007
                        -2.447406481564940e-005
                         1.270798987373922e-004
                         1.535760188691171e-004
```

Hlavne ku koncu, pre väčšie x, je chyba celkom „slušná“. Na obrázku ju nebude vidno, predsa však pri možnostiach matlabu by sme čakali viac. Budeme preto v ďalších príkladoch hľadať spôsoby, ako presnosť zvýšiť.

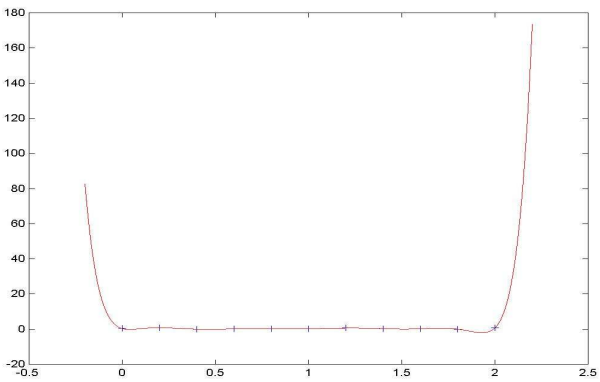
Obrázok:
```
>> plot(x,y,'+'), hold on
>> xh=1:0.001:17; yh=polyval(a,xh);
>> plot(xh,yh,'m')
```



Obrázok je v zásade dobrý, nepresnosti nevidno. Obsahuje však vážny rušivý moment a to je tvar grafu vľavo a vpravo. Hoci graf prechádza určenými bodmi, medzi nimi sa vlní až príliš. Ak chceme interpolačný polynóm použiť na doplnenie hodnôt medzi tými, ktoré sú zadané, asi nebudeme s výsledkami spokojní.

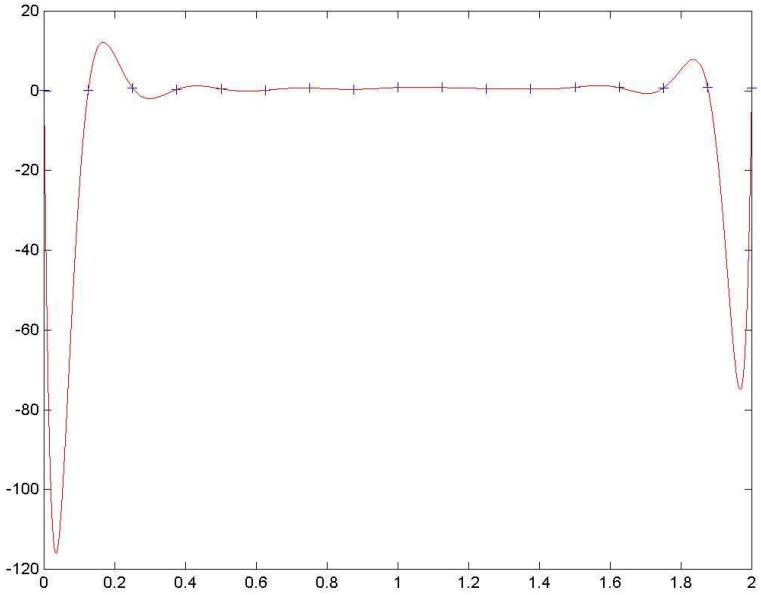
3. Hlavnou príčinou nežiaduceho tvaru grafu na okrajoch je veľký počet bodov x. Sedem bodov sa dalo, ale 17 či dokonca viac začne spôsobovať problémy. Presvedčme sa ešte na jednom príklade, že je to tak. Vezmime si 11 hodnôt x a to v rozsahu 0 až 2, aby ML nehundral, že matica V sa mu zdá takmer singulárna.

```
» x=0:0.2:2; y=rand(1,11); plot(x,y,'+'), hold on
» V=vander(x); a=V\y';
» xh=-0.2:0.0001:2.2; yh=polyval(a,xh); plot(xh,yh,'r')
```



Nie je to najhoršie. Vidíme však, že na okrajoch sa to už začína vlniť. Predĺžme x na 17 hodnôt:

```
» x=0:0.125:2; y=rand(1,17); plot(x,y,'+'), hold on
» V=vander(x); a=V\y';
» plot(x,y,'+'), hold on, xh=0:0.0001:2; yh=polyval(a,xh); plot(xh,yh,'r')
```

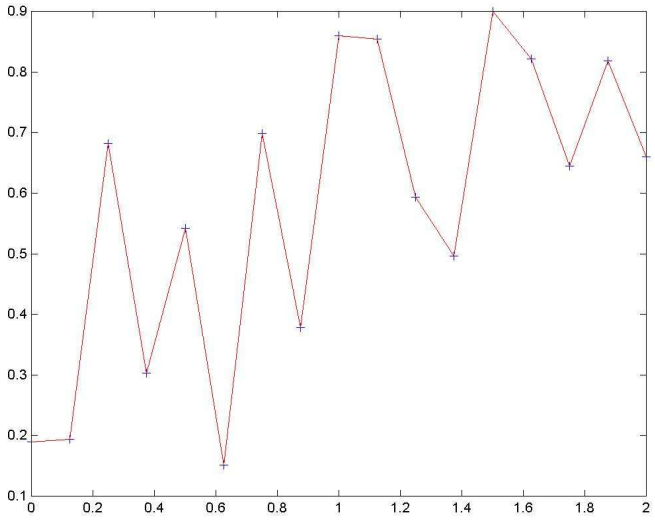


Obrázok potvrdil naše obavy.

4. Riešenie uvedených ťažkostí je jednoduché – nepokúšať sa interpolovať polynómami príliš veľkého stupňa, ak na to nie sú vážne dôvody. Pokiaľ teda máme bodov viac, je vhodné si rozdeliť skúmaný interval na osi x na niekoľko menších podintervalov a získané interpolačné funkcie potom „zlepiť“. Tak sa postupuje napr. pri numerickom integrovaní.

Ostaňme pri zadaní predošlého príkladu. Interpolovať budeme na začiatok polynómami 1. stupňa, teda lomenou čiarou. Postup je rýchly:

```
» plot(x,y,'+', x,y,'r')
```



Postúpme k polynómom 2. stupňa. Budeme osobitne riešiť úlohu pre tieto trojice bodov:

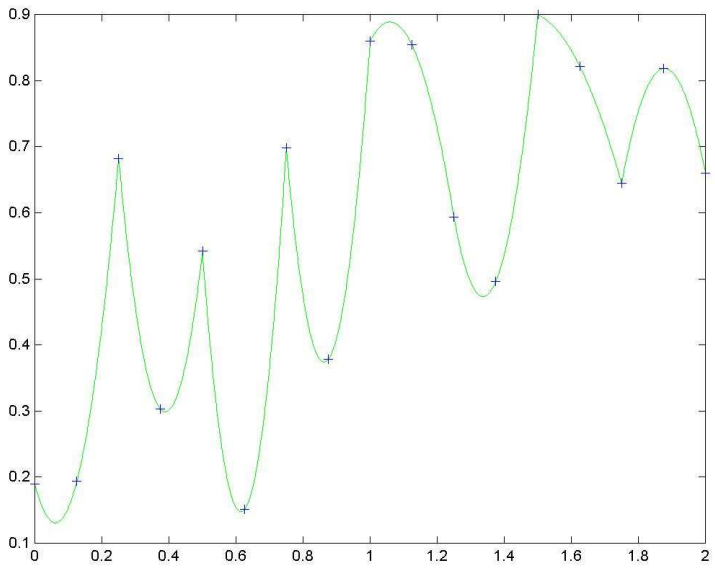
```
» for i=1:8, X(i,:)=x([2*i-1, 2*i, 2*i+1]); end, X

X =
         0    0.1250    0.2500
    0.2500    0.3750    0.5000
    0.5000    0.6250    0.7500
    0.7500    0.8750    1.0000
    1.0000    1.1250    1.2500
    1.2500    1.3750    1.5000
    1.5000    1.6250    1.7500
    1.7500    1.8750    2.0000
```

Vyrobíme si príslušné trojice hodnôt y:

```
» for i=1:8, Y(i,:)=y([2*i-1, 2*i, 2*i+1]); end
```

Napokon počítajme a kreslime interpolačné polynómy na každom intervale:

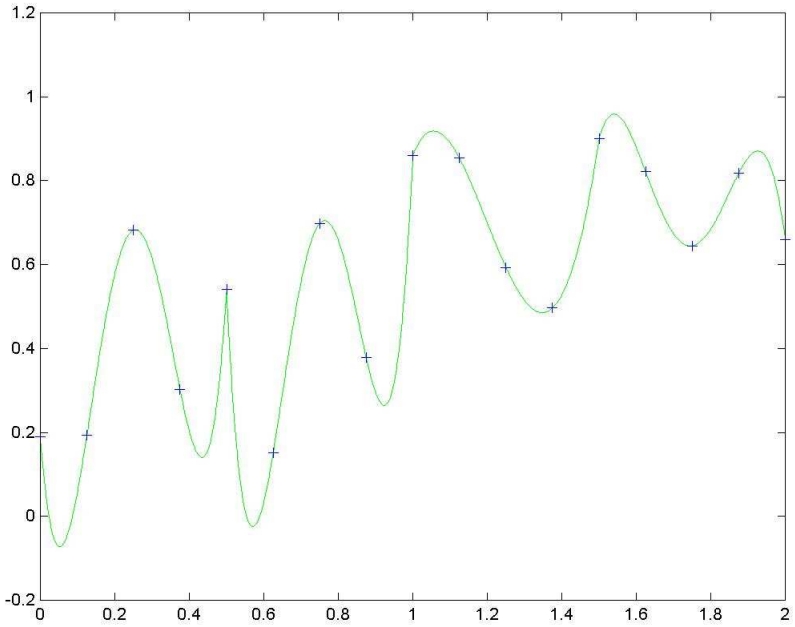
```
» plot(x,y,'+'), hold on
» for i=1:8, V=vander(X(i,:)); a=V\Y(i,:)'; xh=X(i,1):0.001:X(i,3); yh=polyval(a,xh); plot(xh,yh,'g'), hold on, end
```



Je to o čosi lepšie ako lomená čiara, ale tých Kriváňov by mohlo byť menej.

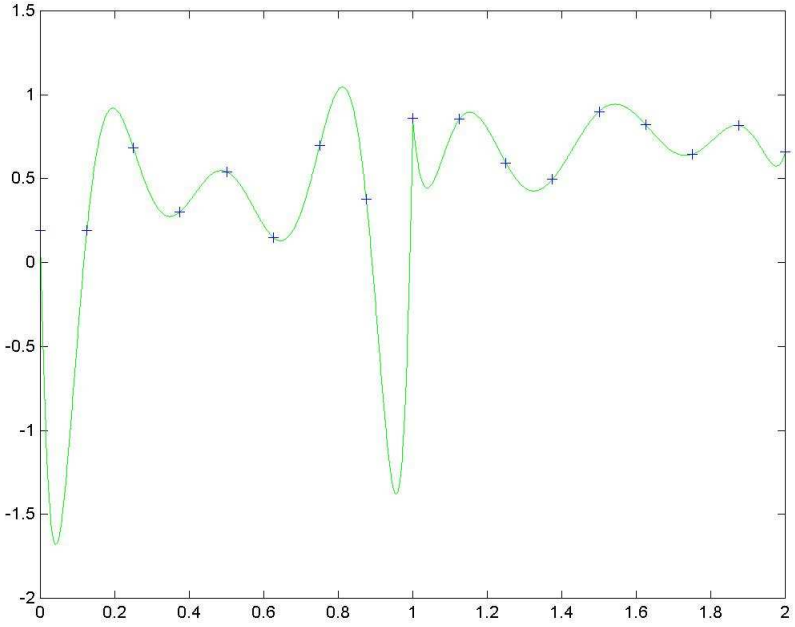
Skúsme štvrtý stupeň interpolačného polynómu:

```
» clear X Y, r=0:4; for i=1:4, q=4*i-3; X(i,:)=x([q+r]); end, X
» for i=1:4, q=4*i-3; Y(i,:)=y([q+r]); end, Y
» for i=1:4, V=vander(X(i,:)); a=V\Y(i,:)'; xh=X(i,1):0.001:X(i,5); yh=polyval(a,xh); plot(xh,yh,'g'), hold on, end
» plot(x,y,'+')
```



Už to vyzerá lepšie. Ukončíme to 8. stupňom, teda riešime v podstate len 2 úlohy.

```
» clear X Y, r=0:8; for i=1:2, q=8*i-7; X(i,:)=x([q+r]); end, X
» for i=1:2, q=8*i-7; Y(i,:)=y([q+r]); end, Y
```



Pravá polovica obrázku je prijateľná, ale vľavo už počet 8 je veľa. Pre 4. stupeň to bolo lepšie. Viac nie je vždy lepšie.

Úloha:

Podobným spôsobom riešte úlohu pre x=0:0.1:6; y=rand(1,61)*10-5. Interpolujte po úsekoch polynómami 1., 2., 3., 4., 5., 6., 10. a 12. stupňa.

Dodatok:

Interpolácia daných bodov po kratších úsekoch polynómami nižších stupňov je postačujúca pri niektorých úlohách ako napr. numerické integrovanie (Newtonove-Cotesove vzorce). Jej vážnym estetickým a niekedy aj matematickým nedostatkom sú spoje jednotlivých polynómov – výsledná interpolujúca funkcia je síce spojitá, jej derivácie však už nie. Tento problém možno úspešne riešiť dômyselnejšou konštrukciou čiastkových interpolačných polynómov. Viac o tom v skriptách (4. kapitola – nepovinná) alebo stručne tu:

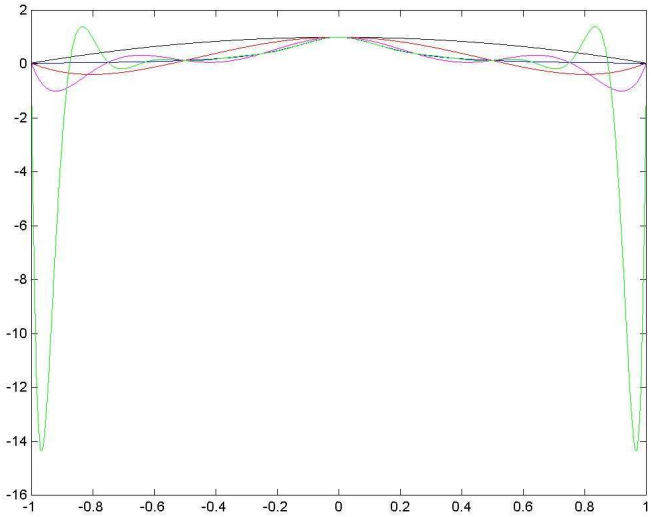
http://en.wikipedia.org/wiki/Interpolation

Stretli sme sa s problémom, že polynómy vyššieho stupňa sa pri krajných bodoch nesprávajú „slušne“. Klasickým príkladom, ktorý ilustruje skrytú zradnosť polynomiálnej interpolácie – a to vôbec nejde o polynómy vysokého stupňa, je tzv. *Rungeho fenomén*.

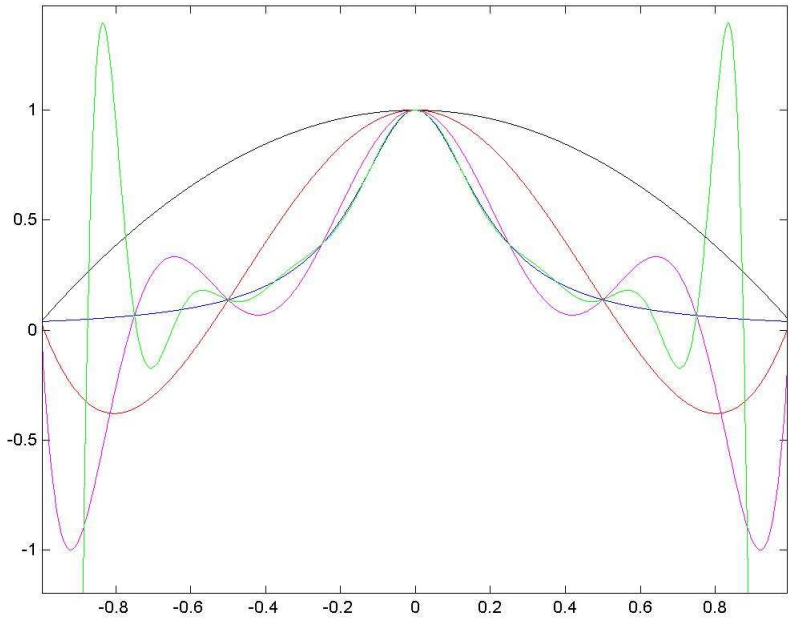
Zvolíme si uzly x na intervale –1, 1 a hodnoty y v týchto uzloch bnude určovať funkcia 1/(1+25x*x) . Začneme s tromi uzlami a budeme delenie intervalu zhusťovať. Necháme si rôznymi farbami vykresliť interpolačné polynómy:

```
» x=-1:0.001:1; y=1./(1+25*x.*x);
» plot(x,y)

» xx=-1:1:1; yy=1./(1+25*xx.*xx);
» V=vander(xx); a=V\yy'; z=polyval(a,x); hold on, plot(x,z,'k')

» xx=-1:0.5:1; yy=1./(1+25*xx.*xx);
» V=vander(xx); a=V\yy'; z=polyval(a,x); hold on, plot(x,z,'r')

» xx=-1:0.25:1; yy=1./(1+25*xx.*xx);
» V=vander(xx); a=V\yy'; z=polyval(a,x); hold on, plot(x,z,'m')

» xx=-1:0.125:1; yy=1./(1+25*xx.*xx);
» V=vander(xx); a=V\yy'; z=polyval(a,x); hold on, plot(x,z,'g')
```

Jednotlivé interpolácie vyzerajú takto (pôvodná funkcia je modrou):



Pri tom zelenom grafe nižšie interpolácie vyzerajú dobre, ale je to klamné zdanie. Zväčšíme si náhľad:



A ešte viac: