Hľadanie interpolačného polynómu

v Newtonovom tvare

Budeme riešiť rovnaké alebo podobné príklady, aké sme už riešili s VanDerMondovou maticou. Očakávame vyššiu presnosť výsledkov.

```
» x=1:7;
» y=rand(1,7)
                y =    0.9501    0.2311    0.6068    0.4860    0.8913    0.7621    0.4565
»
```

Zostavíme si prvý stĺpec hodnôt pomerných diferencií. Vektorové vymoženosti matlabu to umožňujú spraviť naraz:

```
» dy1=(y(2:7)-y(1:6))./(x(2:7)-x(1:6))

dy1 =   -0.7190    0.3757   -0.1209    0.4053   -0.1292   -0.3056
```

Druhý stĺpec:

```
» dy2=(dy1(2:6)-dy1(1:5))./(x(3:7)-x(1:5))

dy2 =    0.5473   -0.2483    0.2631   -0.2673   -0.0882
```

Ďalšie stĺpce:

```
» dy3=(dy2(2:5)-dy2(1:4))./(x(4:7)-x(1:4))

dy3 =   -0.2652    0.1705   -0.1768    0.0597

» dy4=(dy3(2:4)-dy3(1:3))./(x(5:7)-x(1:3))

dy4 =    0.1089   -0.0868    0.0591

» dy5=(dy4(2:3)-dy4(1:2))./(x(6:7)-x(1:2))

dy5 =   -0.0391    0.0292

» dy6=(dy5(2)-dy5(1))./(x(7)-x(1))

dy6 =    0.0114
```

Výsledky máme, ale bolo vcelku otravné každý stĺpec vyrábať zvlášť. Niežeby to nebolo poučné, ale už keď sme teraz poučení, radi by sme dobudúcna tieto výpočty robili rýchlejšie, na jeden príkaz. Už keď je matlab taký šikovný, určite to dokáže.

Tabuľka pomerných diferencií sa dá reprezentovať ako matica. Budeme ju „plniť“ tak, aby číslovanie koeficientov zodpovedalo vyššie uvedeným výpočtom. Stĺpce budeme robiť postupne, lebo na každý z nich potrebujeme ten predošlý. Cyklu sa nevyhneme…

Ak máme 7 riadkov v tabuľke, musíme ísť po 6. diferenciu (ak je riadkov n, diferencie sa počítajú po n-1). Prvý stĺpec je samotný vektor y, ostatné na začiatok vyplníme nulami:

```
» Yd=zeros(7,7); Yd(:, 1)=y';
» for i=1:6, Yd(1:(7-i),i+1)=(Yd(2:(8-i),i)-Yd(1:(7-i),i))./(x((i+1):7)-x(1:(7-i)))'; end, Yd

Yd =

   0.9501   -0.7190    0.5473   -0.2652    0.1089   -0.0391    0.0114
   0.2311    0.3757   -0.2483    0.1705   -0.0868    0.0292         0
   0.6068   -0.1209    0.2631   -0.1768    0.0591         0         0
   0.4860    0.4053   -0.2673    0.0597         0         0         0
   0.8913   -0.1292   -0.0882         0         0         0         0
   0.7621   -0.3056         0         0         0         0         0
   0.4565         0         0         0         0         0         0
```

Z priestorových dôvodov sme nechali na vypisovanie formát so 4 desat. miestami, ale pracovať budeme s čo najpresnejšími hodnotami. Dôležitý je prvý riadok, v ktorom sú koeficienty Newtonovho polynómu:

```
» format long e
» Np=Yd(1,:)

Np =

 Columns 1 through 3

  9.501292851471754e-001  -7.189907715728876e-001   5.473474207701932e-001

 Columns 4 through 6

 -2.652098377233953e-001   1.089166593088519e-001  -3.914529886098098e-002

 Column 7

  1.138841491927743e-002
```

Úloha: Zmeňte indexovanie v cykle tak, aby výsledkom bola dolná trojuholníková matica a aby boli koeficienty Newtonovho polynómu na uhlopriečke.

Dosadzovanie do Newtonovho polynómu si zjednodušíme tak, aby sme nemuseli použiť dva cykly. Využijeme fintu (ilustrujeme na štyroch bodoch), ktorá je známa pod menom Hornerova schéma:

$$a0+a1(x-x1)+a2(x-x1)(x-x2)+a3(x-x1)(x-x2)(x-x3) = a0+(x-x1)(a1+(x-x2)(a2+(x-x3)a3))$$

My máme 7 hodnôt, ale princíp ostáva rovnaký. Výpočet začína „odzadu“. Na začiatok si zistíme, či funkčné hodnoty polynómu naozaj „dosadnú“ do zadaných x,y. Vyčíslime preto hodnoty Newtonovho polynómu priamo v zadaných bodoch x:

```
» u=x; format
» p=Np(7); for i=6:-1:1, p=p.*(u-x(i))+Np(i); end, p

p =   0.9501   0.2311   0.6068   0.4860   0.8913   0.7621   0.4565
```

Takto to vyzerá rovnako ako y, ale to sú len 4 desat. miesta. Pozrime sa na rozdiel pôvodných hodnôt y a funkčných hodnôt:

```
» format long e
» (p-y)'

ans =
                   0
                   0
                   0
  1.110223024625157e-016
  3.330669073875470e-016
  1.110223024625157e-015
  2.331468351712829e-015
```

Je to slušné a aspoň o rád presnejšie než výsledky získané cez VanDerMonda.

Prejdeme na druhý príklad so 17 riadkami v tabuľke hodnôt. Aby sme sa v budúcnosti vyhli prepisovaniu hraníc v cykloch, pri čom sa dá ľahko pomýliť, preformulujeme si príkazy tak, aby sa dali použiť pre ľubovoľné n.

```
 » n=17; x=1:17; y=exp(0.1*x)+rand(1,17);
» Yd=zeros(n,n); Yd(:, 1)=y';
» for i=1:n-1, Yd(1:(n-i),i+1)=(Yd(2:(n+1-i),i)-Yd(1:(n-i),i))./(x((i+1):n)-x(1:(n-i)))'; end, Yd;
```

Výpočet prebehne hladko, matlab si nesťažuje tak ako pri VanDerMondovi. Maticu Yd si tu nedáme vypísať, je dosť veľká. Uspokojíme sa s prvým riadkom:

```
» Np=Yd(1,:)

Np =

  Columns 1 through 3
                   1.123674561323872e+000   9.191353611315511e-001  -5.836915558288885e-001
  Columns 4 through 6
                   2.880542893326951e-001  -9.452341549003890e-002   2.298303349645346e-002
  Columns 7 through 9
                  -4.807846334612889e-003   9.090872412120292e-004  -1.261365264955378e-004
  Columns 10 through 12
                   2.881974465772536e-006   4.316766472074913e-006  -1.438173714411014e-006
  Columns 13 through 15
                   2.923955457897936e-007  -4.612835840478475e-008   6.412063046804922e-009
  Columns 16 through 17
                  -8.830673234077920e-010   1.290012498040575e-010
```

Vyčíslime si funkčné hodnoty v zadaných x a pozrieme sa, nakoľko presne dosadli do zadaných hodnôt y:

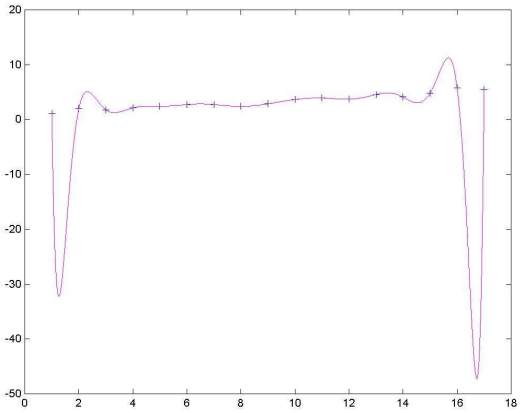
```
» u=x; format
» p=Np(n); for i=(n-1):-1:1, p=p.*(u-x(i))+Np(i); end, p;
» format long e, (p-y)'

ans =
```

```
                   0
                   0
                   0
                   0
-4.440892098500626e-016
-1.776356839400251e-015
-2.664535259100376e-015
-4.440892098500626e-016
                   0
 1.998401444325282e-014
-2.353672812205332e-014
-7.549516567451065e-015
-1.563194018672220e-013
-9.272582701669307e-013
-2.486899575160351e-014
-6.279421427279885e-013
-1.208544375685960e-011
```

Pri vyšších hodnotách x to nie je síce veľká sláva, ale ak si porovnáme túto presnosť s tou, ktorú sme dostali cez VanDerMonda, je to zásadný pokrok o 7 až 9 rádov.

Nakreslíme si obrázok:

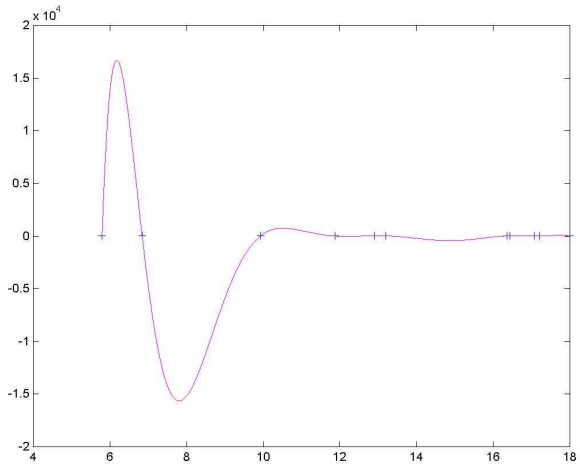
```
» plot(x,y,'+'), hold on
» xh=1:0.001:n; p=Np(n); for i=(n-1):-1:1, p=p.*(xh-x(i))+Np(i); end
» plot(xh,p,'m')
```



Úloha pre Poirota: Prečo tento graf interpolačného polynómu nie je zhodný s tým, ktorý sme dostali pri riešení rovnakej úlohy cez VanDerMonda? Podľa teórie predsa interpolačný polynóm je určený jednoznačne a rôzne postupy sa líšia len tvarom, v akom polynóm nájdu, a v presnosti vyčíslenia koeficientov.

Zatiaľ sme pracovali len s hodnotami x, ktoré boli od seba vzdialené rovnako. Naše vzorce to však nevyžadujú, vedia pracovať aj s hodnotami x rozhádzanými ľubovoľne. Posledný príklad bude o tom. Hodnoty x, y si vyrobíme priamo v stĺpcovom tvare – ušetríme si transpozície.

```
» n=11; x=rand(n,1)*20; x=sort(x); y=rand(n,1)*35;
» Yd=zeros(n,n); Yd(:,1)=y;
» for i=1:n-1, Yd(1:(n-i),i+1)=(Yd(2:(n+1-i),i)-Yd(1:(n-i),i))./(x((i+1):n)-x(1:(n-i))); end, Yd;
» Np=Yd(1,:);
» plot(x,y,'+'), hold on
» xh=x(1):0.001:x(n); p=Np(n); for i=(n-1):-1:1, p=p.*(xh-x(i))+Np(i); end, yh=p;
» plot(xh,yh,'m')
```



Úloha:

Nech x je vektor prvých n (voľte si rôzne n) prvočísel, ktorých cifry sú samy prvočísla (tj. nebude tam 19, 29 apod.):

x = [2 3 5 7 11 13 17 23 31 33 …]

Nech y= rand(1,n).*x .

Nájdite interpolačný polynóm k týmto x, y.