2.6 Някои практически въпроси, свързани с интерполирането с алгебрични полиноми

Числените методи, както казахме, включват голям брой аритметични пресмятания. Затова тяхното прилагане става посредством имплементирането им в компютърни програми. Нека сега дефинираме функция в Mathematica, която намира автоматично полинома на Лагранж по формулата на Нютон. Тъй като в Mathematica индексацията в списъците започва от 1, нека означим интерполационните възли с $x_1,\ldots,x_{n+1}$. Тогава интерполационната формула на Нютон ще изглежда така:

$$L_n(f;x) = f[x_1] + \sum_{k=2}^{n+1} f[x_1,\ldots,x_k](x-x_1)\ldots(x-x_{k-1}).$$

Ясно е, че за да имплементираме формулата на Нютон, трябва да можем да пресмятаме разделени разлики. Вземайки предвид това, ще дефинираме следните функции:

- $dividedDiff[nodes\_, values\_]$ – изчислява разделената разлика на функция със стойности $values$ в точките $nodes$ ($nodes$ и $values$ се задават като списъци);
- $newtonPoly[nodes\_, values\_, x\_]$ – намира интерполационния полином на Лагранж, построен за възли $nodes$ и стойности $values$.

```
dividedDiff[nodes_, values_] := (
  If[Length[nodes] == 1,
   values[[1]],
   (dividedDiff[nodes[[2 ;; Length[nodes]]], values[[2 ;; Length[nodes]]]] -
      dividedDiff[nodes[[1 ;; Length[nodes] - 1]], values[[1 ;; Length[nodes] - 1]]]) /
    (nodes[[Length[nodes]]] - nodes[[1]])]
 )
newtonPoly[nodes_, values_, x_] := (
  poly = 0;
  product = 1;
  For[i = 1, i ≤ Length[nodes], i++,
   poly += dividedDiff[nodes[[1 ;; i]], values[[1 ;; i]]] * product;
   product *= x - nodes[[i]]
  ];
  poly // Expand
 )
```

Сега, използвайки дефинираните функции, ще разгледаме някои особености на интерполирането с алгебрични полиноми, които трябва да се имат предвид, когато приближаваме дадена функция или моделираме дадено явление. Ще започнем с една дефиниция.

Дефиниция 1. *Когато използваме интерполационния полином за приближаване на стойност в интервала, определен от интерполационните възли, говорим за* ***интерполация****. В противен случай говорим за* ***екстраполация****.*

Задача 1. В таблицата са дадени данни за населението на САЩ в периода 1920-1990. Да се построи полином от седма степен, интерполиращ таблицата. Да се даде приближение на населението през 1952, 1974, 2000 година и да се сравни с действителните стойности – съответно 157 млн., 214 млн., 281.42 млн.

Година	1920	1930	1940	1950	1960	1970	1980	1990
Население	106.46	123.08	132.12	152.27	180.67	205.05	227.23	249.46

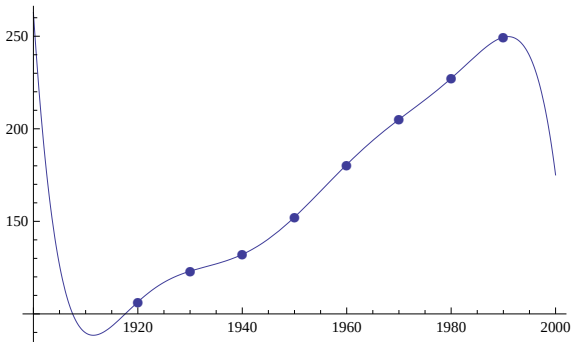
Решение. Ще използваме програмата в Mathematica, която предложихме в началото на настоящия параграф.

```
In[18]:= nodes = Range[1920, 1990, 10];
         values = {106.46, 123.08,  132.12, 152.27, 180.67, 205.05, 227.23, 249.46};
         p[x_] := newtonPoly[nodes, values, x];
         p[1952]
         p[1974]
         p[2000]
Out[21]= 157.728
Out[22]= 213.511
Out[23]= 175.08
```

Получихме следните приближения:

- През 1952 година – 157.728 млн. (в действителност 157 млн.)
- През 1974 година – 213.511 млн. (в действителност 214 млн.)
- През 2000 година – 175.08 млн. (в действителност 281,42 млн.)

В първите два случая приближението е добро и относителната грешка е малка. В третия обаче полученият с интерполационния полином резултат няма нищо общо с действителността. Отново виждаме, че при екстраполация не можем да разчитаме на добри резултати. Обърнете внимание как в границите на интерполация (между 1920 и 1990) поведението му добре моделира нарастването на населението, а извън тях това не е така.



□

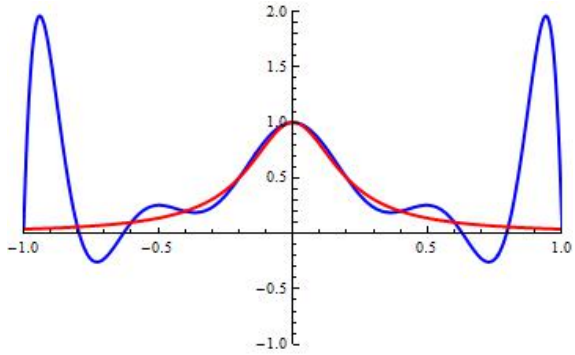
Със следващата задача ще покажем, че, противно на интуицията, често при налагане на повече интерполационни условия (съответно използвайки полиноми от по-висока степен) получаваме по-лошо приближение. Това е така, защото полиномите от по-висока степен имат „по-лошо" поведение – появяват се осцилации.

Задача 2. Да се приближи функцията[1] $f(x) = \dfrac{1}{1+25x^2}$ в интервала, като се намерят интерполационните полиноми от степени 10 и 4 при равноотдалечени възли в интервала.

Да се построят графиките на абсолютните грешки в двата случая.

Решение. Първо да намерим полинома от десета степен.

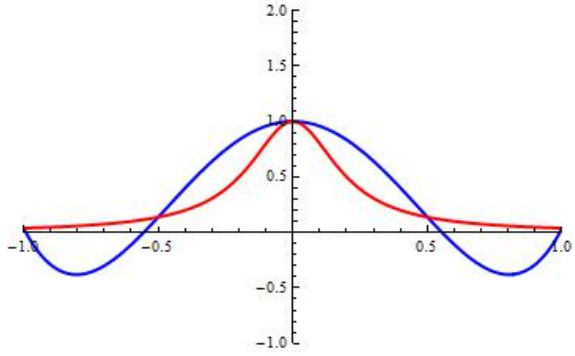
```
In[85]:= f[x_] := 1/(1 + 25 x^2)
         nodes = Range[-1, 1, 0.2];
         values = Table[f[x], {x, -1, 1, 0.2}];
         p[x_] = newtonPoly[nodes, values, x]
         Plot[{p[x], f[x]}, {x, -1, 1},
          PlotStyle → {{Blue, Thick}, {Red, Thick}}, PlotRange → {{-1, 1}, {-1, 2}}]
```



При интерполиране с полиноми от висока степен можем да очакваме наличието на осцилации. Между възлите на интерполация поведението на интерполационния полином е „лошо" - виждате как в двата края на интервала грешката при апроксимация е много голяма.

Приближението (като цяло) е по - добро с полинома от по - ниската степен:

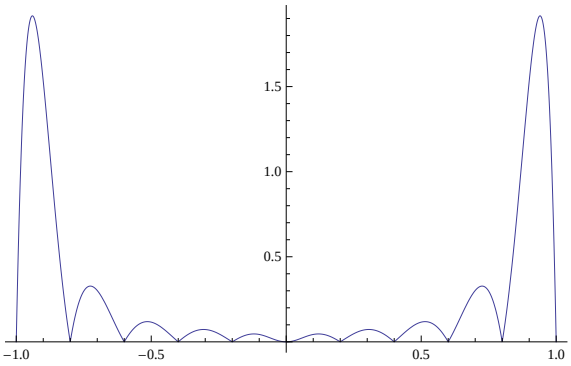
```
In[91]:= f[x_] := 1/(1 + 25 x^2)
         nodes = Range[-1, 1, 0.5];
         values = Table[f[x], {x, -1, 1, 0.5}];
         p4[x_] = newtonPoly[nodes, values, x]
         Plot[{p4[x], f[x]}, {x, -1, 1},
          PlotStyle → {{Blue, Thick}, {Red, Thick}}, PlotRange → {{-1, 1}, {-1, 2}}]
```



[1] Това е т.нар. функция на Рунге, носеща името на Карл Рунге – немски математик, забелязал особеността при интерполиране с полиноми от висока степен, която коментираме. Тази особеност е известна в литературата като „феномен на Рунге".

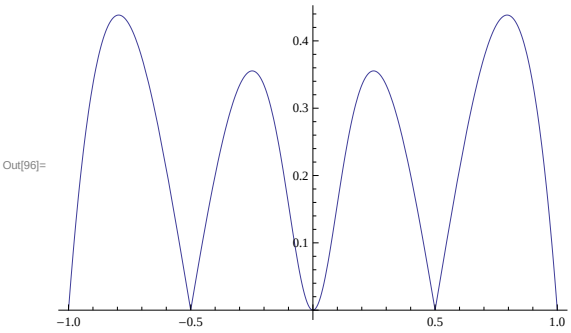
Нека сега построим графиките на абсолютните грешки при апроксимацията с полиномите от десета и четвърта степен. За полинома от десета степен получаваме следната графика:

```
In[90]:= Plot[Abs[f[x] - p[x]], {x, -1, 1}, PlotRange → All]
```



За полинома от четвърта степен имаме:

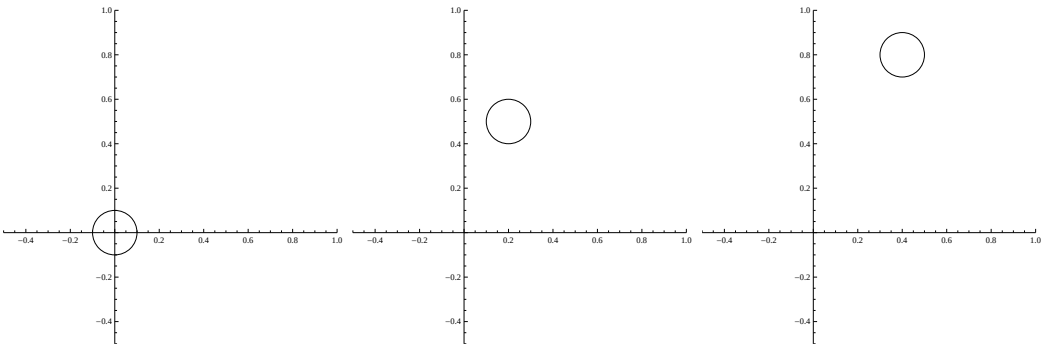
```
In[96]:= Plot[Abs[f[x] - p4[x]], {x, -1, 1}, PlotRange → All]
```



Вижда се, че и в двата случая грешката в средата на интервала е по-малка отколкото в краищата. Това особено добре си личи за полинома от десета степен. Затова е добре, когато интерполираме, възлите да са подбрани така, че точката, в която искаме да намерим приближената стойност, да е близо до средата на интервала. □

Нека сега видим как можем да направим една простичка анимация.

Задача 3. Дадени са три сцени:



Да се направи анимация на базата на тези три сцени

Решение. Можем да намерим функция, която описва траекторията на центъра на окръжността. За целта можем да построим полином от втора степен (защото имаме три точки), който минава през точките (0,0), (0.2,0.5), (0.4,0.8) (това е центърът на окръжността на трите сцени). Решавайки тази интерполационна задача, можем да рисуваме окръжността през достатъчно малки интервали от време с център върху тази парабола и така ще получим анимирано движение на окръжността.

```
p[x_] = newtonPoly[{0, 0.2, 0.4}, {0, 0.5, 0.8}, x];
Animate[Graphics[Circle[{x, p[x]}, 0.1], Axes → True,
  AxesOrigin → {0, 0}, PlotRange → {{-0.5, 1}, {-0.5, 1}}],
  {x, 0, 0.4, 0.01}]
```

□

Задача 4. Проведени са експерименти, за да се определи бързодействието на един алгоритъм за сортиране в зависимост от броя елементи. Резултатите са представени в следната таблица:

бр.е/ти (x1000)	10	20	50	100	150	200	250
време, сек	0.1639275	0.53282	3.00007	11.20784	26.7486723	47.3297	76.80605

Да се определи колко най-много елемента могат да се сортират за не повече от 30сек.

Решение. Нека с x означим броя елементи. Ще използваме **обратна интерполация**. Т.е. ще построим интерполационния полином $f(x)$, съответсващ на таблицата, и ще намерим стойността на x, за която $f(x) = 30$. От проведените експерименти е ясно, че търсената стойност за x ще е между 150 и 200 хил. Нека построим интерполационния полином от трета степен с възли 100, 150, 200, 250 и стойности – 11.20784, 26.7486723, 47.3297, 76.80605. По този начин точката, в която искаме да апроксимираме стойността, ще има по два възела от двете си страни.

```
p[x_] = newtonPoly[{100, 150, 200, 250},
   {11.20784, 26.7486723, 47.3297, 76.80605}, x];
Solve[p[x] == 30, x]
```

```
Out[103]= {{x → 47.4033 - 243.128 i}, {x → 47.4033 + 243.128 i}, {x → 159.083}}
```

Получаваме $x \approx 159.083$. Следователно за под 30 сек. можем да сортираме масив с приблизително 159 хил. елемента. □

Библиография

[1] Боянов, Б.: Лекции по числени методи. Дарба, 2008

[2] Сборник по числени методи – http://www.fmi.uni-sofia.bg/econtent/nummeth

[3] Сендов, Бл., Попов, В.: Числени методи. Първа част. Университетско издателство „Св.Климент Охридски", 1996

[4] Kiusalaas, J.: Numerical Methods in Engineering. Cambridge University Press, 2010

[5] Chapra, S.: Applied Numerical Methods with Matlab for Engineers and Scientists. McGraw Hill, 2012

[6] Бахвалов, Н.С., Лапин, А.В., Чижонков, Е.В.: Численные методы в задачах и упражнениях. Высшая школа, 2000

[7] Hollis, S: Manual for Stewart's Single Variable Calculus. Brooks/Cole, 2008

[8] Antia, H. M.: Numerical Methods for Scientists and Engineers. McGraw-Hill Publishing Company Ltd., 1991