Erik Dalsryd, Patrick Henning
2018-11-20

**SF1511, Numeriska metoder och grundläggande programmering**
**– Laboration 2 –**
**Polynomanpassning**

Deadline - 2018-12-07

## Introduktion

*Efter den här laborationen ska du kunna skriva lite större program som är uppdelade i egendefinierade funktioner.*

# Polynomanpassaren

Du ska skriva ett program som kan användas för att hantera polynom av godtycklig grad. Användaren ska via en meny kunna mata in ett gradtal och ett antal punkter som polynomet ska anpassas till. Punkterna måste vara fler än gradtalet, kontrollera det. Är de en fler ritas polynomet som går exakt genom dessa punkter. Är de ännu fler ritas det med minsta kvadrat-metoden anpassade polynomet. Sedan, då användaren ber om det, ska polynomets koefficienter beräknas och polynomet och punkterna ritas upp. Dessutom ska användaren kunna få reda på vilka nollställen polynomet har.

Så här ska det se ut när användaren anger gradtal och punkter:

```
----- Polynomanpassaren -----
1) Ange gradtal
2) Ange punkter
3) Beräkna koefficienter
4) Plotta polynomet
5) Hitta nollställen
0) Avsluta
Ditt val: 1
Vilket gradtal ska polynomet ha? 2
----- Polynomanpassaren -----
1) Ange gradtal
2) Ange punkter
3) Beräkna koefficienter
4) Plotta polynomet
5) Hitta nollställen
0) Avsluta
Ditt val: 2
Ange punkter (retur avslutar inmatningen)
Ange en punkt ([x y]): [1 1]
Ange en punkt ([x y]): [2 1]
Ange en punkt ([x y]): [3 2]
Ange en punkt ([x y]):
```

Du ska själv skriva en funktion för varje menyval utom »Avsluta». Så här ska de anropas från huvudprogrammet:

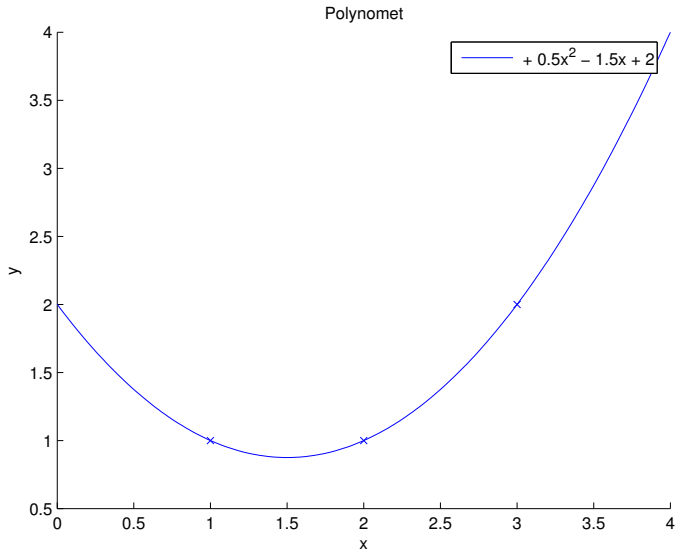
| | | |
|---|---|---|
| `degree = askdegree();` | – | Frågar efter, läser in och returnerar gradtal. |
| `points = askpoints();` | – | Frågar efter, läser in och returnerar punkterna. |
| `coeffs = findcoeffs(degree, points);` | – | Löser det linjära (överbestämda) ekvationssystemet och returnerar polynomets koefficienter. Du får inte använda `polyfit`; se nedan om minsta kvadrat-po |
| `plotpoly(coeffs, points);` | – | Ritar upp polynomet och punkterna. |
| `polyroots(coeffs);` | – | Bestämmer och skriver ut polynomets nollställen. |

Utöver funktionerna ovan är det vettigt att skriva en funktion som bara skriver ut menyn. Skriver du programmet på det sättet blir koden i huvudprogrammet mer överskådlig. Då du anropar menyfunktionen i huvudprogrammet kan det se ut så här:

```
meny();
choice = input('Ditt val: ');
```

Då användaren ber om att koefficienterna ska beräknas ska de också skrivas ut. Innan dess måste han/hon ha angett gradtal och punkter.

Innan programmet ritar upp grafen bör det ha beräknat koefficienterna.
Grafen för exemplet ovan blir:




## Tips, trix och krav

- Förslag på arbetsordning:
  1. Börja med att konstruera tomma funktioner som inte gör annat än skriver ut sitt namn.
  2. Konstruera menyn och anropa för respektive val de tomma funktionerna. Inget ska hända när användaren väljer något, utom att programmet avslutas när man väljer Avsluta.
  3. Gör färdigt funktionerna i tur och ordning.
  4. När du är färdig med de två första funktionerna är det skönt att temporärt lägga in värden på gradtal och punkter direkt i koden, så man slipper skriva in det varje gång resten av funktionerna ska provas. Kom ihåg att ändra tillbaka sedan!

- Grafen:
  - Använd `plot` för att rita polynomet. Programmet bör rita ut polynomet över ett intervall som täcker in punkterna. Ta fram min och max för punkternas x-värden! Från en vektor av x-värden beräknas y-värdena med `polyval` eller en `for`-slinga.

    *Frivilligt*: gör sådan »legend» som syns i exempelgrafen.
- Använd `roots` för att beräkna nollställena.
- Som avbrottsvillkor vid inmatning av punkter kan du använda att det inmatade har en längd som är skild från två (punkterna är ju vektorer med längden två).
- Du får inte använda några globala variabler!
- Du får inte avbryta while-loop med break.

*Lycka till!*

## Om minsta-kvadrat-polynom

### Linjära minsta-kvadrat-problem i `Matlab`

Vi börjar med ett linjärt ekvationssystem

$$Ax = b$$

där koefficientmatrisen $A$ är $m \times n$ och $x$ är en $n$-kolonnvektor med komplexa (eller reella) element. $b$ är en $m$-kolonnvektor. Om $m = n$ och matrisen $A$ inte är singulär finns precis en lösning som vi skriver (men inte beräknar) som $x = A^{-1}b$. Om $m > n$ är systemet *överbestämt* och det finns i allmänhet ingen exakt lösning. Vi söker då ett $x$ som gör residualvektorn (eller felvektorn) $r = b - Ax$ minst i den meningen att summan av kvadraterna på felen

$$\sum_{k=1}^{m} r_k^2$$

minimeras. Man kan visa, att lösningen till detta problem satisfierar *normalekvationerna*

$$A^T Ax = A^T b$$

I `Matlab` löser man linjära ekvationssystem med bakåtdivisions (backslash)-operatorn `x = A \ b;` och `Matlab` använder olika algoritmer i de två fallen. Mer därom senare i numme-kursen.

### Minsta-kvadrat-polynom i `Matlab`

Låt nu $m$ punkter $(x_i, y_i), i = 1, \ldots, m$ vara givna, så att inga två $x$ är lika. Vi söker ett polynom av gradtal $N$

$$p(x) = \sum_{j=1}^{N+1} c_j x^{N+1-j} = c_1 x^N + c_2 x^{N-1} + \ldots + c_N x + c_{N+1}$$

vars graf går så nära alla punkterna som möjligt. Vi skriver polynomet såhär eftersom `Matlab`s polynom-funktioner `polyfit`, `polyval` och `roots` använder denna ordning av koefficienterna $c_i$. Vi formulerar vårt önskemål som ett överbestämt linjärt ekvations-system

$$p(x_i) = y_i, i = 1, 2, \ldots, m : \sum_{j=1}^{N+1} c_j x_i^{N+1-j} = y_i$$

dvs.

$$Ac = y$$

med $A$ så att

$$a_{ij} = x_i^{N+1-j}, i = 1, \ldots, m, j = 1, 2, \ldots, N+1$$

$A$'s kolonn $j$ har alltså potensen $N+1-j$ av $x$-värdena. Låt kolonn-vektorerna `x` och `y` vara givna. Då kan vi konstruera matrisen $A$ kolonn för kolonn:

```
A = zeros(m,N+1);
for k = 1:N+1
   A(:,k)=x.^(N+1-k);
end;
```

och beräkna koefficienterna i minsta-kvadrat-lösningen med `\`:

```
c = A \ y;
```

Övertyga dig själv så att du kan förklara vid redovisningen att detta $c$ minimerar

$$\sum_{i=1}^{m} (p(x_i) - y_i)^2$$