Politechnika Wrocławska

Wielomiany i Aproksymacja

Wojciech Myszka

17 listopada

HR EXCELLENCE IN RESEARCH

Politechnika Wrocławska

1. Trend
2. Wielomiany
3. Aproksymacja
4. Interpolacja

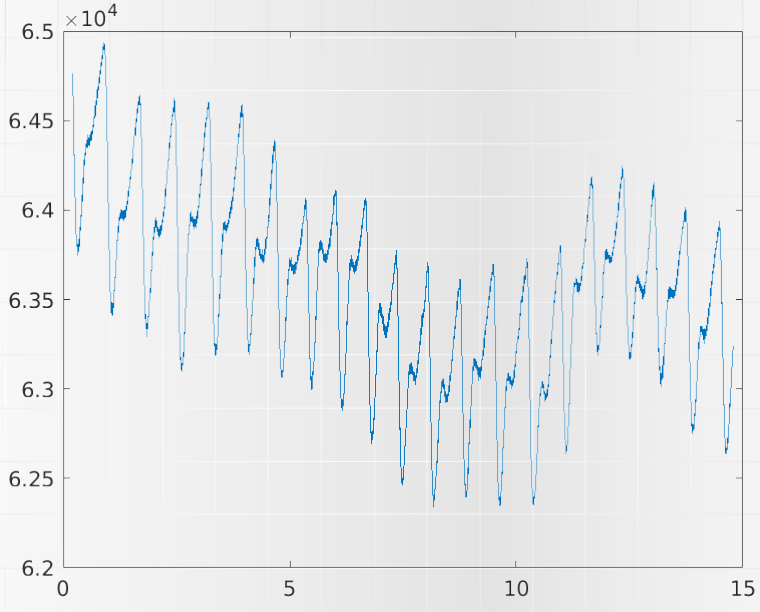
Politechnika Wrocławska

Trend

Likwidacja trendu I

1. Bardzo często pomiary zaburzone są w następujący sposób:

Likwidacja trendu II



Politechnika Wrocławska

Funkcja `detrend()` I

1. Jednym z narzędzi dostarczanych przez MATLAB jest funkcja `detrend()` służąca do usuwania trendu wielomianowego.
2. Zakłada ona, że analizowany przebieg nałożony jest na wielomian stosunkowo niskiego stopnia.
3. Wywołanie funkcji:
   - `y = detrend(x)` — najprostszy przypadek, trend liniowy
   - `y = detrend(x,n)` — wielomian stopnia `n`
   - `y = detrend(x,n,bp)` — pozwala dodać punkty, w których funkcja trendu się będzie zmieniać.

Funkcja `detrend()` II

4. Dodatkowe informacje na temat trendu na stronie projektu.

Wielomiany

Politechnika Wrocławska

Wielomiany I

1. Wielomiany to wyrażenia postaci

$$y = \sum_{i=1}^{n} p_i x^{n+1-i} = p_1 x^n + p_2 x^{n-1} + \cdots + p_n x + p_{n+1}$$

2. Wektor p zawiera współczynniki wielomianu. **Zwracam uwagę na ich kolejność i numerację!** (n to rozmiar tablicy p)
3. Wielomian $x^2 + 3x - 5$ opisany jest wektorem `[1, 3, -5]`
4. Funkcja `polyval()` może być użyta do wyznaczenia wartości wielomianu

Wielomiany II

```
p = [1, 3, -5];
x = -3:3;
polyval(p, x)
ans =
    -5    -7    -7    -5    -1     5    13
```

5. Funkcja `roots()` wylicza pierwiastki wielomianu

```
roots(p)
ans =
   -4.1926
    1.1926
```

6. Funkcja `poly()` pozwala wygenerować współczynniki wielomianu znając jego pierwiastki:

Wielomiany III

```
poly(ans)
ans =
    1.0000    3.0000   -5.0000
```

7. Funkcje `polyint()` i `polyder()` służą do całkowania i różniczkowania wielomianów:

```
polyint(p)
ans =
    0.3333    1.5000   -5.0000         0
polyder(p)
ans =
     2     3
```

8. Iloczyn wielomianów realizowany jest przez funkcję `conv()` Prosty przykład:

Politechnika Wrocławska

Wielomiany IV

```
>> a=poly([1,2])
a =
     1    -3     2
b=poly([3, 4])
b =
     1    -7    12
conv(a, b)
ans =
     1   -10    35   -50    24
roots(ans)
ans =
    4.0000
    3.0000
```

Wielomiany V

```
2.0000
1.0000
```

Sprawdzenie poprawności obliczeń pozostawiam słuchaczom.

Uwaga: Nazwa funkcji (`conv`) pochodzi od słowa *convulsion* co tłumaczy się na polski jako *splot* (dwu funkcji) i wtraża się wzorem:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-\tau)g(\tau)d\tau$$

W przypadku wielomianów sprowadza się to do iloczynu.

Wielomiany VI

9. Dzielenie wielomianów wykonywane jest przez funkcję `deconv()`. Pamiętać trzeba, że zazwyczaj podczas dzielenia wielomianów powstaje tak zwana reszta (z dzielenia); wywołanie funkcji jest następujące:

   ```
   [q, r] = deconv(u, v)
   ```

   `q` to wynik dzielenia, `r` to reszta (zazwyczaj też wielomian)

   Przykład

   ```
   u = [1 -10 35 -50 24];
   v = [1 -7 12];
   ```

   (dane pochodzą z poprzedniego przykładu)

Wielomiany VII

```
[q, r] = deconv(u, v)
q =
     1    -3     2
r =
     0     0     0     0     0
```

czyli podzieliło się bez reszty (czego można było oczekiwać); nie będzie tak w ogólnym przypadku

```
u = [2 7 4 9];
v = [1 0 1];
[q,r] = deconv(u,v)
q = 1×2
     2     7
r = 1×4
```

Wielomiany VIII

```
     0     0     2     2
```

sprawdzenie

```
conv(v, q)
ans =
     2     7     2     7
```

teraz dodamy resztę

```
ans+r
ans =
     2     7     4     9
```

czyli działa...

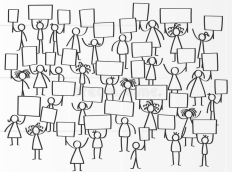
Wielomiany IX

10. Suma i różnica — nie ma gotowych funkcji: trzeba wyrównać długość obu wektorów parametrów uzupełniając krótszy zerami z lewej strony i, po prostu, dodać/odjąć je; później, być może, trzeba zlikwidować nadmiarowe zera z lewej strony.

Politechnika Wrocławska

Wielomiany symboliczne

1. MATLAB jest wyposażony w *Symbolic Math Toolbox*
2. Pozwala on na prowadzenie obliczeń w sposób symboliczny (a nie numeryczny).
3. Polecenie `syms` deklaruje które obiekty są symboliczne



Aproksymacja

Politechnika Wrocławska

Co to jest? I

1. Aproksymacja to inaczej przybliżanie zbioru par punktów jakąś funkcją.
2. Jeżeli punkty pochodzą z obserwacji jakiegoś zbioru losowego — jest to, po prostu, **regresja**.
3. Inaczej niż w *interpolacji*, nie żąda się aby funkcja przechodziła przez punkty pomiarowe.
4. Oczekuje się, że będzie przechodziła **najbliżej** tych punktów.
5. Bardzo często podczas aproksymacji szukamy takiej funkcji $f(x)$, że dla danego zestawu danych $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ wartość

$$Q = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

Co to jest? II

była **najmniejsza**

6. Jest to **aproksymacja średniokwadratowa**.

Politechnika Wrocławska

Przybliżenie krzywej

1. W MATLABie dostępna jest w pakiecie **Curve Fitting Toolbox** funkcja `fit()`
   - `fitobject = fit(x,y,fitType)` — podstawowe użycie (jeden wymiar)
   - `fitobject = fit([x,y],z,fitType)` — podstawowe użycie (dwa wymiary)
   - `fitobject = fit(x,y,fitType,fitOptions)` — dodatkowe opcje
   - `fitobject = fit(x,y,fitType,Name,Value)` — opcje definiowane jako pary *nazwa, wartość*
   - `[fitobject,gof] = fit(x,y,fitType)` — dodatkowe informacje na temat jakości przybliżenia
   - `[fitobject,gof,output] = fit(x,y,fitType)` — jeszcze więcej informacje
2. O funkcji `polyfit()` będzie później.

Dostępne funkcje przybliżenia I

1. `poly1`, `poly2`,..., `poly9` — wielomiany (jednej zmiennej), aż do 9 stopnia
2. `poly`*ij* — wielomiany dwu zmiennych, *i* oraz *j* oznaczają stopień wielomianu wobec każdej zmiennej
3. `weibull` — funkcja Weibulla ($Y = abx^{(b-1)} \exp(-ax^b)$)
4. `exp1`, `exp2` — funkcje wykładnicze ($Y = a\exp(bx)$ oraz $Y = a\exp(bx) + c\exp(dx)$)
5. `fourier1`, `fourier2`, ..., `fourier9` — wielomiany trygonometryczne, aż do stopnia 9; podstawowy okres wynika z zakresu zmienności zmiennej x
6. `gauss1`, `gauss2`, ..., `gauss8` (jednowymiarowe) wielomiany postaci:
$$Y = \sum_{i=1}^{n} a_i \exp(-\left(\frac{x - b_i}{c_i}\right)^2) \qquad 1 \leq n \leq 8$$
7. `power1`, `power2` — $Y = ax^b$ oraz $Y = ax^b + c$

Dostępne funkcje przybliżenia II

8. `rat`ij — ilorazy wielomianów i stopień licznika, j stopień mianownika
9. `sin1`, `sin2`,..., `sin8` — $Y = \sum_{i=1}^{n} a_i \sin(b_i x + c_i) \qquad 1 \leq n \leq 8$
10. `cubicspline`, `smothingspline` — splajny

Użycie (na przykładzie) I

(census to przykładowy zbiór danych statystycznych)

```
load census;
f=fit(cdate,pop,'poly2')
```

W wyniku powstaje obiekt f (typu cfit)

```
f =
     Linear model Poly2:
     f(x) = p1*x^2 + p2*x + p3
     Coefficients (with 95% confidence bounds):
       p1 =    0.006541  (0.006124, 0.006958)
       p2 =      -23.51  (-25.09, -21.93)
       p3 =   2.113e+04  (1.964e+04, 2.262e+04)
```

Użycie (na przykładzie) II

Przedziały ufności (*confidence bounds*) mają sens tylko wtedy, gdy o danych można założyć, że mają rozkłady (normalne) prawdopodobieństwa.

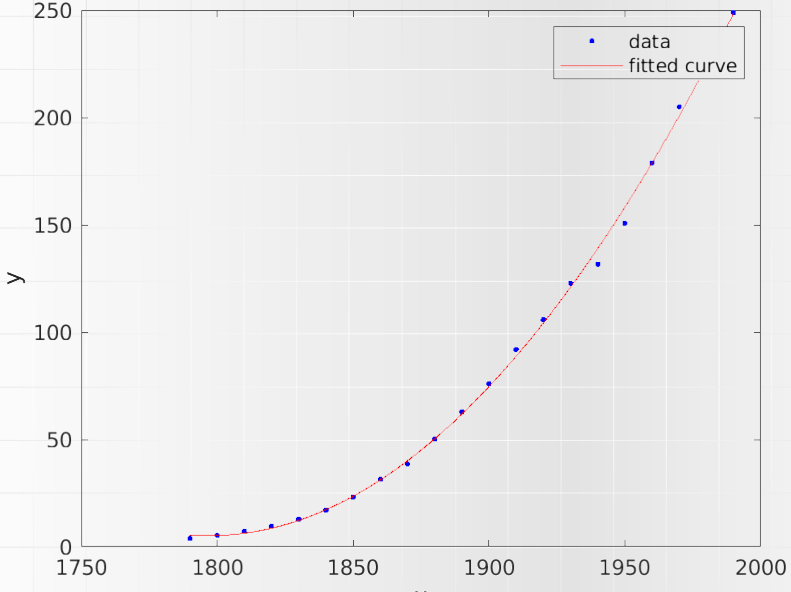
Dostęp do współczynników jest naturalny

```
>> f.p1
ans =
    0.0065
```

Można też rysować wykresy

```
plot(f,cdate,pop)
```

Użycie (na przykładzie) III



Politechnika Wrocławska

Przybliżanie funkcji wielomianami

1. Funkcja `polyfit()` wyznacza współczynniki wielomianu najlepiej przybliżającego zestaw punktów $\{(x_i, y_i), \quad i = 1, \ldots, n\}$
2. Użycie:

   ```
   p = polyfit(x, y, n)
   ```

   `x` i `y` to wektory zawierające współrzędne punktów; `n` stopień wielomianu.

Własne modele I

1. Funkcja `fittype()` pozwala konstruować własne modele używane do aproksymacji.
2. Załóżmy, że interesuje nas model postaci

$$Y = ax + b\sin(x) + c$$

3. Stworzymy go w sposób następujący

```
>> ft = fittype({'x','sin(x)','1'})
ft =
     Linear model:
     ft(a,b,c,x) = a*x + b*sin(x) + c
```

Własne modele II

4. Model jest **liniowy**, gdyż współczynniki (a, b, c) wchodzą do modelu liniowo.
5. Możemy użyć go w obliczeniach w sposób następujący

   generujemy dane

   ```
   >> x=1:10;
   >> y=2*x+3;
   ```

   (zwracam uwagę, że y nie uwzględnia funkcji *sinus*; zmienne x i y trzeba będzie transponować!)

Własne modele III

```
>> fit(x',y',ft)
ans =
     Linear model:
     ans(x) = a*x + b*sin(x) + c
     Coefficients (with 95% confidence bounds):
       a =           2  (2, 2)
       b =   7.201e-16  (-2.974e-15, 4.415e-15)
       c =           3  (3, 3)
```

współczynnik b jest (zgodnie z oczekiwaniami) równy zero. Ze względu na brak zaburzeń przedziały ufności są zerowej szerokości.

Politechnika Wrocławska

Własne modele IV

Uwaga

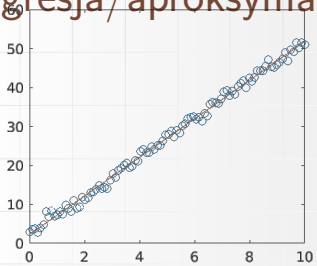
Warto zapoznać się z dokumentacją pakietu *Curve Fitting Toolbox*, żeby rozpoznać wszystkie jego możliwości.

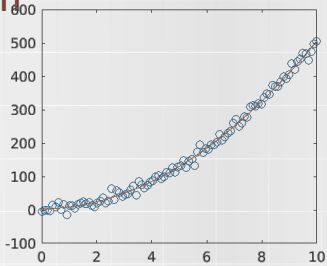
Politechnika Wrocławska

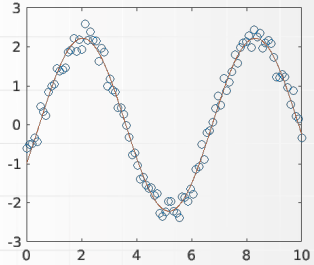
Regresja/aproksymacja I

1. Używamy funkcji MATLABa `lscov()`
2. Dane: M par punktów $\{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}$ zapisane w zmiennych `x` i `y`
3. Regresja liniowa $a_1 + a_2 x$:
   - 3.1 budujemy macierz `F = [ones(M,1), x']`
   - 3.2 `a = lscov(F, y)`
4. Regresja wielomianowa $a_1 + a_2 x + a_3 x^2 + \cdots + a_{n+1} x^n$
   - 4.1 budujemy macierz `F = [ones(M,1), x', (x^2)',..., (x^n)']`
   - 4.2 `a = lscov(F, y)`
5. Regresja postaci: $a_1 + a_2 f_1(x) + a_3 f_2(x) + \cdots + a_{n+1} f_n(x)$
   - 5.1 budujemy macierz `F = [ones(M,1), (f1(x))', (f2(x))',...,(fn(x))']`
   - 5.2 `a = lscov(F, y)`

Regresja/aproksymacja II







Politechnika Wrocławska

Interpolacja

Interpolacja I

Załóżmy że mamy (zmierzone lub wyliczone) wartości jakiejś funkcji w kilku(nastu) punktach. Dla uproszczenie niech będzie to funkcja $\sin(x)$ w punktach co $\pi/4$ w zakresie od 0 do 2π.

```
x = 0:pi/4:2*pi;
v = sin(x);
```

Chcemy na tej podstawie wyznaczyć wartość funkcji **pomiędzy** tymi punktami. Dla uproszczenia przyjmijmy, że będą to punkty pomiędzy 0 a 2π z krokiem $\pi/16$.

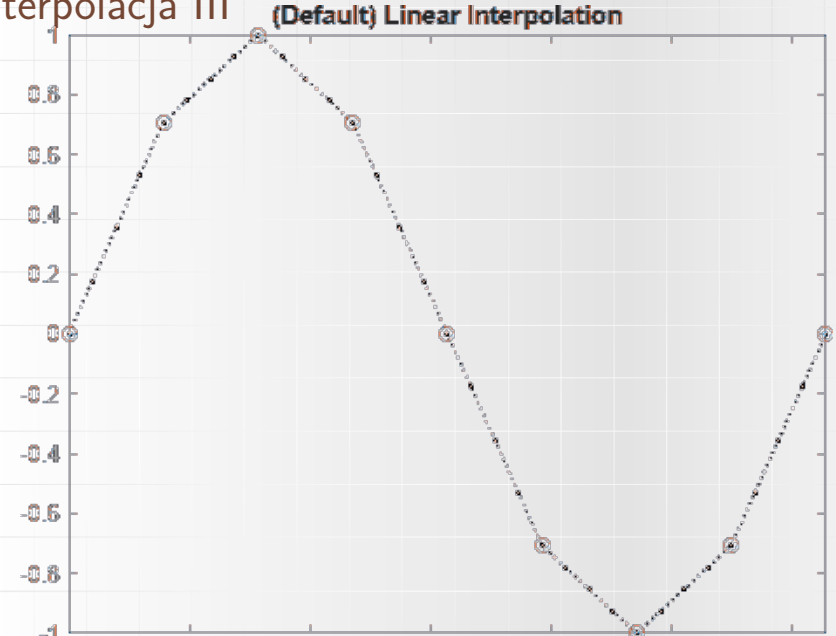
Takie zadanie nazywane bywa **interpolacją**.

Najprościej wyznaczać wartość z punktów „pośrednich" używając przybliżenia liniowego.

Interpolacja II

```
figure
vq1 = interp1(x,v,xq);
plot(x,v,'o',xq,vq1,':.');
xlim([0 2*pi]);
title('(Default) Linear Interpolation');
```

Interpolacja III



Politechnika Wrocławska

Interpolacja vs aproksymacja