Approssimazioni ai minimi quadrati

Consideriamo di avere un certo numero di misurazioni "y" per certi tempi "x", rappresentiamone un grafico:

```
>> x = [ 0:19];
    x =
          0   1   2   3   ...   17   18   19
>> y= [ 6 13 23 33 54 65 79 96 112 145 186 253 305 ...
       402 523 785 1063  1125  1370 1575 1830];
>> plot(x,y)
```

attraverso il grafico é possibile visualizzare come si dispongono i punti, il quale ad esempio

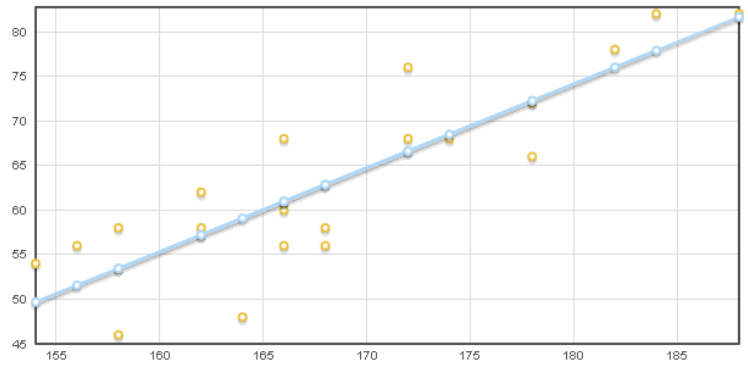

Figure 1: Confronto tra dati reali e approssimazione lineare

puó essere quello dell'andamento di una certa funzione, come una legge *lineare, esponenziale, parabolica, logaritmica*. Tale problema si risolve attraverso il cosí detto metodo ai **"Minimi quadrati"**.

Consideriamo una legge lineare [1]

$$y = mx + q \qquad \text{o} \qquad y - mx - q = 0, \qquad \text{con m e q parametri incogniti.} \tag{1}$$

Dato l'insieme di misurazioni $(x_i, y_i), i = 0, ..., n$, se ne calcolano i rispettivi valori in 1,

$$y_i - mx_i - q$$

se ne fa il quadrato e la somma

$$\varphi = \sum_{i=0}^{n} (y_i - mx_i - q)^2$$

φ é una funzione dipendente da m e q, vogliamo derminare la retta tale che $\varphi(m, q)$ sia minima, essendo φ una funzione strettamente convessa, il valore

$$\min_{m,q} \varphi(m, q)$$

é raggiunto per i valori di m e q tali che

$$\begin{cases} \frac{\partial \varphi}{m} = 0 \\ \frac{\partial \varphi}{q} = 0 \end{cases}$$

Andiamo a vedere come si risolve il problema in MatLab

[1] Rivedere i fondamenti in Matlab per i polinomi Introduzione al linguaggio Matlab (polinomi)

```
>> p1 =polyfit(x,y,1);  % calcola il polinomio di grado 1
                        % che approssima meglio i dati
    p1 =
              89.8541       -316.314
```

confrontiamo i grafici tra i dati sperimentali e la retta calcolata:

```
>> plot(x,y,'o',x,polyval(p1,x))   % il comando 'o' disegna dei pallini
```

Se dal grafico si nota che l'approssimazione attraverso una retta non é buona (i punti non si dispongono su una funzione lineare), ma se si nota che il grafico ricorda maggiormante quello di una parabola, applicheremo il metodo ai minimo quadrati per un polinomio di grado 2:

$$y = ax^2 + bx + c,$$

in questo caso si dovranno determinare i valori di a, b, c ottimi:

```
>> p2=polyfit(x,y,2)   %  approssimazione mediante un polinomio di grado 2

  p2 =
         6.656   -36.624   63.120

>> y2=polyval(p2,x);
>> plot(x,y,'o',x,y2);
```

Nota: É possibile aumentare il grado del polinomio a piacimento.

É importante visulizzare gli errori commessi aumentando il grado del polinimio

```
>> y1=polyval(p1,x);
>> r1=y-y1;
>> r2=y-y2;
>> r3=y-y3;
>> plot(x,r1);
```

Nota: Se l'andamento dell'errore é irregolare (oscillante) alloral'errore é casuale e l'approssimazione é buona.

Esempio: Approssimazioni con polinomi di grado qualsiasi.

```
>> x=[1:9];
>> y=[5 6 10 20 26 33 34 36 42];
>> plot(x,y) % e' una spezzata
% approssimo con polinomi di grado i = 1,2,3,4... a scelta
>> pi=polyfit(x,y,i);
>> yi=polyval(pi,x);
>> plot(x,y,'o',x,y1)  % considero la retta
% se voglio il grafico della funzione piu' regolare basta aumentare i nodi
>> xx = [1:0.05:9];
>> y4=polyfit(p4,xx);
>> plot(x,y,'o',xx,y4)
```

Ulteriori tipologie di approssimazioni

Andiamo a vedere altre approssimazioni funzionali.

1. $y = \alpha x^{\beta}$,

2. $y = \alpha e^{\beta x}$,
3. $y = \alpha \log x + \beta$,
4. $y = \frac{1}{\alpha x + \beta}$.

É possibile notare che, calcolando le funzioni sui dati speriemenatali, attraverso delle trasformazioni esse sono tutte riconducibili ad approssimazioni lineari:

1. $y_i \approx \alpha x_i^{\beta}$, si passa ai logarimi

$$\begin{aligned} \log y_i &= log(\alpha x_i^{\beta}) \\ &= \log(\alpha) + log(x_i^{\beta}) \\ &= \log(\alpha) + \beta \log(x_i) \end{aligned}$$

   e attraverso la sostituzione

$$\begin{aligned} \xi_i &= \log(x_i) \\ \eta_i &= \log(y_i) \\ a &= \log(\alpha) \\ b &= \beta \end{aligned}$$

   si ottiene la legge lineare

$$\eta_i \approx a + b\xi_i,$$

   si applica quindi il metodo ai minimi quadrati per l'insieme di punti $(\xi_i, \eta_i), i = 0, ..., n$:

```
>> polyfit(log(x), log(y), 1)
```

2. $y = \alpha e^{\beta x}$,

$$\begin{aligned} \log y_i &= log(\alpha e^{\beta}) \\ &= \log(\alpha) + \beta x_i \log(e) \\ &= \log(\alpha) + \beta x_i \end{aligned}$$

   si ottiene la legge lineare

$$\eta_i \approx a + b\xi_i,$$

   si applica quindi il metodo ai minimi quadrati

```
>> polyfit(x,log(y), 1)
```

3. $y = \alpha \log x + \beta$,

$$y_i \approx \alpha \log(x_i) + \beta$$

```
>> polyfit(log(x), y, 1)
```

4. $y = \frac{1}{\alpha x + \beta}$,

$$\frac{1}{y_i} \approx \alpha x_i + \beta$$

```
>> polyfit(x, 1./y, 1)
```

Riassumiamo gli esempi per i comandi grafici:

```
>> plot(x,log(y),'o')
>> plot(log(x),log(y),'o')
>> plot(log(x),y,'o')
>> plot(x,1'/y,'o')
```

da questi grafici si nota qual'é il piú simile ad una retta (ad esempio il primo), se ne determinano allora m e q per i minimi quadrati

```
>> [m q] = polyfit(x,log(y),1)
```

ma ricordiamo che

$$\log y_i \approx mx_i + q$$

e quindi

$$y_i = e^{mx_i+q} = e^{mx_i}e^q$$

indicati con

$$e^q = \alpha \qquad m = \beta$$

bisogna determinare α e β

```
>> p1 = polyfit(x,log(y),1);
>> p1

           -0.45800       1.78971
>> alfa = exp(p1(2))
     alfa =
               5.98651
>> beta=p1(1)
     beta =
              -0.45800
>> xx = [ 0: 0.15:05];
>> yy = alfa*exp(beta*xx);
>> plot(x,y,'o',xx,yy)
```