MINIMI QUADRATI *

A. SOMMARIVA † E M.VENTURIN ‡

Si digiti sulla shell di Matlab/Octave

```
>> x=0:0.01:2*pi;
>> y=sin(2*x)+(10^(-1))*rand(size(x));
>> plot(x,y,'r-');
```

Dal grafico si capisce che la funzione può essere interpretata come una *perturbazione* della funzione $\sin(2x)$ nell'intervallo $[0, 2\pi]$. Ci interessa approssimare non tanto la funzione plottata bensì $\sin(2x)$ (che in qualche modo è la funzione senza *rumore*).
Osserviamo che non ha senso utilizzare un interpolante polinomiale p di grado N nè una spline interpolante visto che *ricostruirebbero* la funzione *perturbata*.
Scriviamo sulla shell di Matlab/Octave `help polyfit`. In una recente release di Matlab appare

```
 POLYFIT Fit polynomial to data.
    POLYFIT(X,Y,N) finds the coefficients of a polynomial P(X)
    of degree N that fits the data, P(X(I))~=Y(I), in a least
    -squares sense.
```

In altri termini `polyfit` calcola i coefficienti del polinomio p_N di grado N che *meglio* approssima (in norma 2 discreta, da cui il nome di *minimi quadrati discreti*) la funzione f avente nel vettore di nodi X i valori Y (cioè $Y(i) := f(X(i))$). Questo punto è delicato [7].
In pratica si cerca il polinomio p_N tale che è minima la quantità

$$\|f - P_N\|_{2,d} = \sqrt{\sum_i |f(x_i) - P_N(x_i)|^2}.$$

Digitiamo quindi in un file `minimiquadrati.m`

```
a=0; b=2*pi;
h=0.01;
x=a:h:b;
y=sin(2*x)+(10^(-1))*rand(size(x));

for n=2:8
    coeff=polyfit(x,y,n); % VALUTAZIONE COEFF.
                          % MIGLIORE APPROX. ("M.A.").

    z=polyval(coeff,x);   % VALORE M.A. NEI NODI "x".
    err2=norm(z-y,2);
    errinf=norm(z-y,inf);
```

*Ultima revisione: 31 maggio 2010

†DIPARTIMENTO DI MATEMATICA PURA ED APPLICATA, UNIVERSITÀ DEGLI STUDI DI PADOVA, VIA TRIESTE 63, 35121 PADOVA, ITALIA (ALVISE@MATH.UNIPD.IT)

‡DIPARTIMENTO DI INFORMATICA, UNIVERSITÀ DEGLI STUDI DI VERONA, STRADA LE GRAZIE 15, 37134 VERONA, ITALIA (MANOLO.VENTURIN@GMAIL.COM)

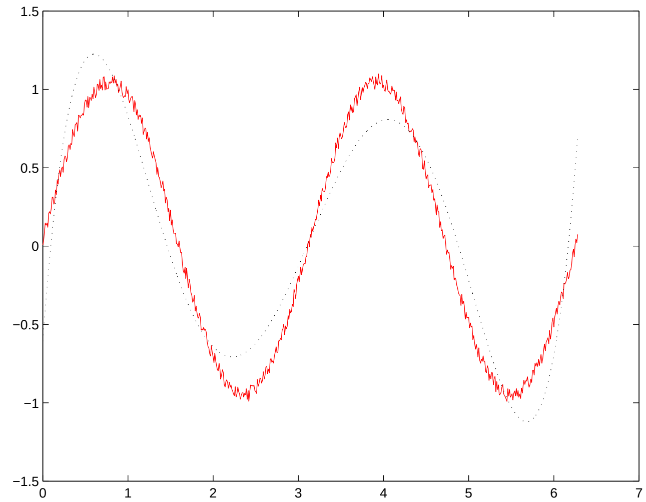

FIGURA 1.1. *Grafico che illustra l'approssimazione ai minimi quadrati di grado 5 su una perturbazione della funzione* $\sin(2x)$ *(campionamento in nodi equispaziati)).*

```
    u=0:(h/4):2*pi;      % NODI TEST.
    v=polyval(coeff,u);  % VALORE M.A. IN NODI TEST.

    plot(x,y,'r-',u,v,'k:');
    fprintf('\n \t [GRADO]: %2.0f [ERR. 2]: %2.2e',n,err2);
    fprintf(' [ERR. INF]: %2.2e',errinf);
    pause(5);
end
```

Osserviamo che

1. `polyval` valuta nei nodi (di test) `u` il polinomio avente coefficienti `coeff` (ordinati dal grado n fino al grado 0);
2. il comando `pause` mette l'esecuzione in pausa per qualche secondo (5 nel nostro caso); la continua successivamente.

Per calcolare gli errori assoluti commessi abbiamo valutato non solo $\|f - P_N\|_{2,d}$ ma pure

$$\|f - P_N\|_{\infty,d} := \max |f(x_i) - P_N(x_i)|.$$

Dal punto di vista numerico se il grado è basso, in virtù delle concavità e delle convessità di $\sin(2\,x)$, il polinomio p_N fornisce un'approssimazione scadente, mentre migliora per $N \geq 5$. Osserviamo che per $N > 10$ tale routine incorre in problemi di condizionamento e stampa un warning del tipo

```
Warning: Polynomial is badly conditioned. Remove repeated data
         points or try centering and scaling as described in
         HELP POLYFIT.
```

Vediamo ora i risultati:

```
>> minimiquadrati
```

```
[GRADO]:  2 [ERR. 2]: 1.63e+001 [ERR. INF]: 1.16e+000
[GRADO]:  3 [ERR. 2]: 1.50e+001 [ERR. INF]: 1.17e+000
[GRADO]:  4 [ERR. 2]: 1.50e+001 [ERR. INF]: 1.17e+000
[GRADO]:  5 [ERR. 2]: 5.48e+000 [ERR. INF]: 6.48e-001
[GRADO]:  6 [ERR. 2]: 5.48e+000 [ERR. INF]: 6.45e-001
[GRADO]:  7 [ERR. 2]: 1.23e+000 [ERR. INF]: 1.56e-001
[GRADO]:  8 [ERR. 2]: 1.23e+000 [ERR. INF]: 1.51e-001
```

Ricordiamo che in generale, non ha senso cercare un grado troppo alto del polinomio *di miglior approssimazione* p_N in quanto si otterrebbe a partire da un certo valore il polinomio interpolante, mentre un grado troppo basso, come già detto, non ricostruirebbe adeguatamente l'andamento della funzione f.

2. **Facoltativo: Un esempio.** Si calcoli il polinomio di grado 3 che meglio approssimi

$$f(x) = \exp(x)$$

arrotondato a 2 cifre decimali, nel senso dei minimi quadrati per

$$x_n = -1 + \frac{n-1}{10},\ n = 1, \ldots, 21.$$

Risoluzione. Un semplice plot mostra in Matlab/Octave il grafico della funzione da approssimare. Osserviamo che l'intervallo in cui effetture l'analisi è $[-1, +1]$ essendo i punti x_n in questo intervallo

```
>> n=1:21;
>> -1+(n-1)/10
ans =
  Columns 1 through 7
   -1.0000   -0.9000   -0.8000   -0.7000   -0.6000   -0.5000   -0.4000
  Columns 8 through 14
   -0.3000   -0.2000   -0.1000         0    0.1000    0.2000    0.3000
  Columns 15 through 21
    0.4000    0.5000    0.6000    0.7000    0.8000    0.9000    1.0000
>>
```

Per quanto riguarda il grafico

```
>> h_test=1/100;
>> h_test=1/100;
>> x_test=-1:h_test:1;
>> y_test_esatto=exp(x_test);
>> y_test=round(y_test_esatto*100)/100;
>> format long
>> y_test'
ans =
   0.37000000000000
   0.37000000000000
   0.38000000000000
   0.38000000000000
   0.38000000000000
```

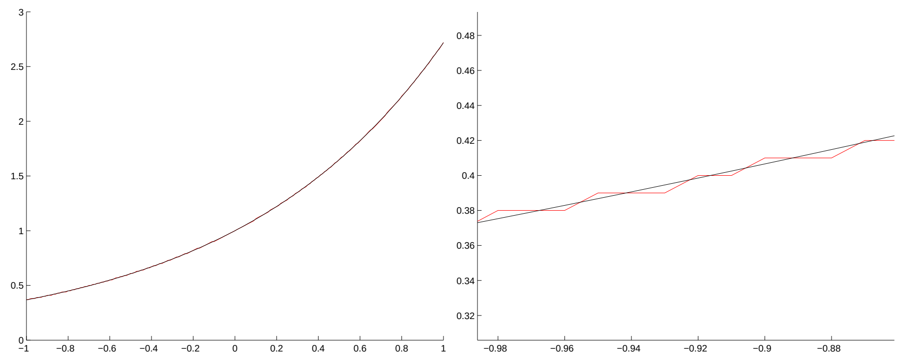

FIGURA 2.1. *Grafico che illustra un confronto tra l'esponenziale (in nero) e l'esponenziale arrotondato a due cifre decimali (in rosso) e un suo zoom intorno a* -0.92.

```
   ... ... ... ...
   2.61000000000000
   2.64000000000000
   2.66000000000000
   2.69000000000000
   2.72000000000000
>> hold on;
>> plot(x_test,y_test,'r-');
>> plot(x_test,y_test_esatto,'k-');
>> hold off;
```

ottenendo il grafico in figura (si noti anche lo zoom intorno a -0.92).

Osserviamo che il comando

```
>> y_test_esatto=exp(x_test);
>> y_test=round(y_test_esatto*100)/100;
```

effettua come richiesto l'arrotondamento dell'esponenziale a due cifre decimali.
Calcoliamo ora il polinomio di terzo grado richiesto.

```
>> n=1:21;
>> xn=-1+(n-1)/10;
>> fxn_esatto=exp(xn);
>> fxn=round(fxn_esatto*100)/100;
>> grado=3;
>> coeffs=polyfit(xn,fxn,grado)
coeffs =
   0.17875166616585   0.54269071889209   0.99536841664988   0.99529911735861
>> x_test=-1:1/100:1;
>> px_test=polyval(coeffs,x_test);
>> hold on;
>> plot(x_test,px_test,'r-');
>> plot(xn,fxn,'ko')
>> hold off
>>
```

Dal grafico successivo si evince quanto accurata sia l'approssimazione. Per ulteriori dettagli si veda [4, p.265]

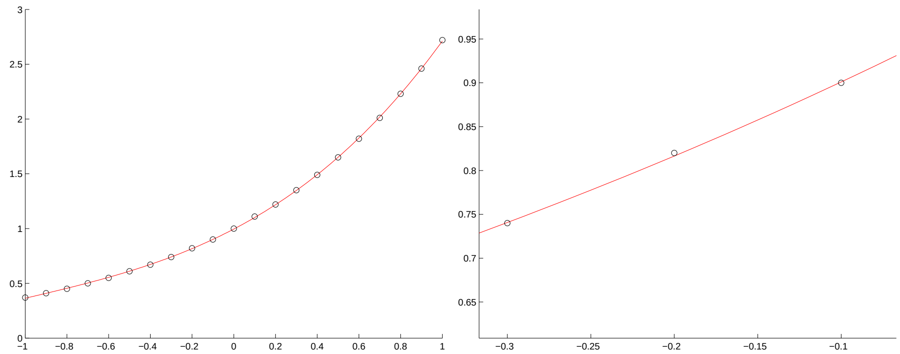

FIGURA 2.2. *Grafico che illustra un confronto tra il polinomio approssimante l'esponenziale arrotondato a due cifre decimali (in rosso) e i punti x_n, e un suo zoom intorno a -0.2.*

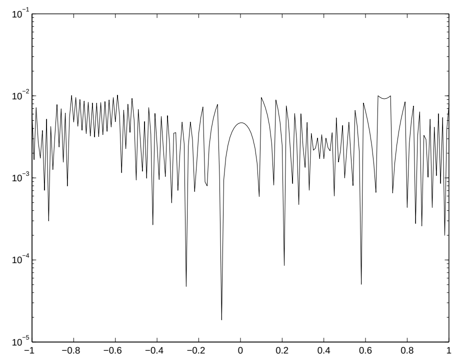

FIGURA 2.3. *Grafico che illustra l'errore assoluto compiuto tra il polinomio approssimante l'esponenziale arrotondato a due cifre decimali (in rosso) nei punti test $x_{test} = -1 : (1/100) : +1$.*

3. Qualche nota storica. Non è molto chiaro chi sia stato lo scopritore del metodo dei minimi quadrati [9]. In generale si da' credito a Gauss, il quale nel 1795 a 19 anni, ebbe tale idea che utilizzò in seguito nel 1801 per predirre la locazione dell'asteroide Cerere. L'astronomo Piazzi dopo aver scoperto Cerere ne perse la traccia visto l'orbita dietro il Sole, e Gauss fu capace di predirne il moto senza passare per le equazioni di Keplero permettendo all'astronomo ungherese Franz Xaver von Zach di ritrovarlo nella posizione indicata da Gauss.

Gauss non pubblicò il metodo fino al 1809 quando apparve nel secondo volume del suo lavoro *Theoria Motus Corporum Coelestium in sectionibus conicis solem ambientium.*

L'idea dei minimi quadrati fu pubblicata indipendentemente dal francese Adrien-Marie Legendre nel 1805 e dall'americano Robert Adrain nel 1808.

4. Online. *Per ulteriori delucidazioni si considerino le seguenti pagine web*

1. *http://it.wikipedia.org/wiki/Minimi_quadrati*
2. *http://it.wikipedia.org/wiki/Minimi_Quadrati*
3. *http://www.dsm.units.it/%7Ebellen/calcolo_numerico/CAPIT-4.PDF*
4. *http://www.mathworks.com/moler/leastsquares.pdf*

5. *http://en.wikipedia.org/wiki/Least_squares*

RIFERIMENTI BIBLIOGRAFICI

[1] *K. Atkinson, Introduction to Numerical Analysis, Wiley, 1989.*

[2] *Bellen, homepage personale,*
http://www.dsm.units.it/%7Ebellen/calcolo_numerico/CAPIT-4.PDF.

[3] *V. Comincioli, Analisi Numerica, metodi modelli applicazioni, Mc Graw-Hill, 1990.*

[4] *S.D. Conte e C. de Boor, Elementary Numerical Analysis, 3rd Edition, Mc Graw-Hill, 1980.*

[5] *The MathWorks Inc.,*
http://www.mathworks.com/moler/leastsquares.pdf.

[6] *The MathWorks Inc., Numerical Computing with Matlab,*
http://www.mathworks.com/moler.

[7] *A. Quarteroni e F. Saleri, Introduzione al calcolo scientifico, Springer Verlag, 2006.*

[8] *A. Suli e D. Mayers, An Introduction to Numerical Analysis, Cambridge University Press, 2003.*

[9] *Wikipedia, Least Squares,*
http://en.wikipedia.org/wiki/Least_squares.

[10] *Wikipedia, Minimi quadrati,*
http://it.wikipedia.org/wiki/Minimi_quadrati.

[11] *Wikipedia, Minimi Quadrati,*
http://it.wikipedia.org/wiki/Minimi_Quadrati.