Serie 5 - Soluzione
Approssimazione di Funzioni e Dati

Data la relativa lunghezza della soluzione proposta e la ripetitività di molti comandi, si consiglia di scrivere la soluzione in uno script (*M-file*) tramite un editor.

Esercizio 1

Si consideri la funzione:

$$f(x) = x\,\sin(x).$$

1. il grafico della funzione nell'intervallo $[-2, 6]$ si può ottenere con i seguenti comandi Matlab® :

```
fun = inline('x .* sin(x)','x');
a = -2;  b = 6;
x_dis = linspace( a, b, 1000 );
f_dis = fun(x_dis);
figure(1)
plot( x_dis, f_dis, 'r')
xlabel('x')
ylabel('y')
title('f(x)=x sin(x)')
```

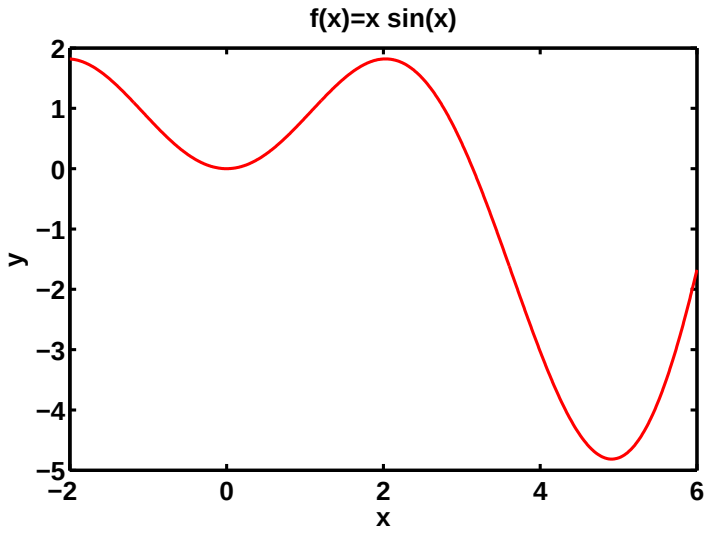

Figura 1: $f(x) = x\,\sin(x)$

In Figura 1 è riportato il grafico tracciato da Matlab® .

2. I polinomi interpolanti di Lagrange $\Pi_n f$, relativi ad una distribuzione di nodi equispaziati, possono essere ottenuti tramite un ciclo `for` che iteri sui gradi richiesti ($n = 2, 4, 6$):

```
PP_dis = [];
err_dis = [];
err_max = [];

for n=2:2:6
    h = ( b - a ) / n;
    x_nod = [ a : h : b ];
    f_nod = fun(x_nod);
    P = polyfit( x_nod, f_nod, n )
    P_dis = polyval( P, x_dis );
    PP_dis = [ PP_dis; P_dis ];
    err_dis = [ err_dis;  abs( P_dis - f_dis ) ];
    err_max = [ err_max; max( abs( P_dis - f_dis ) ) ];
end
```

Le ultime tre righe all'interno del ciclo salvano i risultati appena calcolati, che altrimenti sarebbero sovrascritti all'iterazione successiva. In questi casi è buona norma inizializzare i vettori `PP_dis`, `err_dis` e `err_max` prima del ciclo. L'ultima riga prima di `end` riguarda il calcolo della norma infinito dell'errore, richiesta dall'ultimo punto dell'esercizio.

È possibile sovrapporre il grafico del polinomio interpolante di grado due a quello della funzione $f(x)$ tracciato al punto precedente:

```
hold on
plot( x_dis, PP_dis(1,:),'g')
legend('y=x*sin(x)','pol2')
title('f(x)=x*sin(x) e polinomio interpolatore di secondo grado')
```

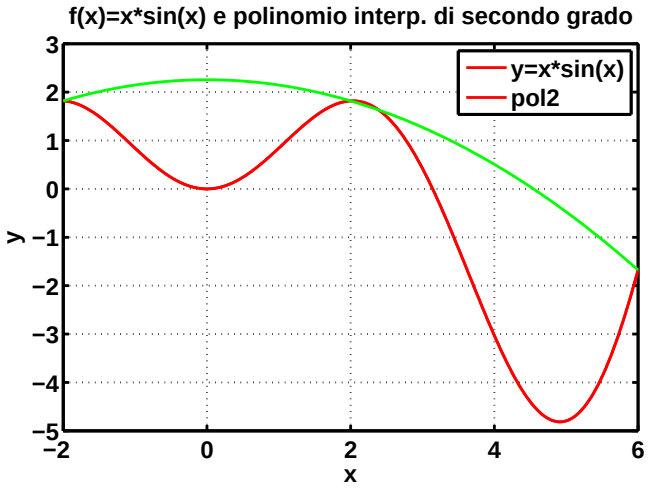

Figura 2: $f(x) = x\sin(x)$ e polinomio interpolatore di secondo grado

Il grafico corrispondente è riportato in Figura 2.

Per rappresentare graficamente la funzione insieme a tutti i polinomi interpolanti calcolati è possibile utilizzare il seguente codice:

```
figure(2)
plot( x_dis, f_dis, 'k')
hold on
grid on
plot( x_dis, PP_dis)
legend('y=xsin(x)','pol 2','pol 4','pol 6')
xlabel('x')
ylabel('y')
title('f(x)=xsin(x) e polinomi interpolanti')
```

Questi comandi producono il grafico di Figura 3.

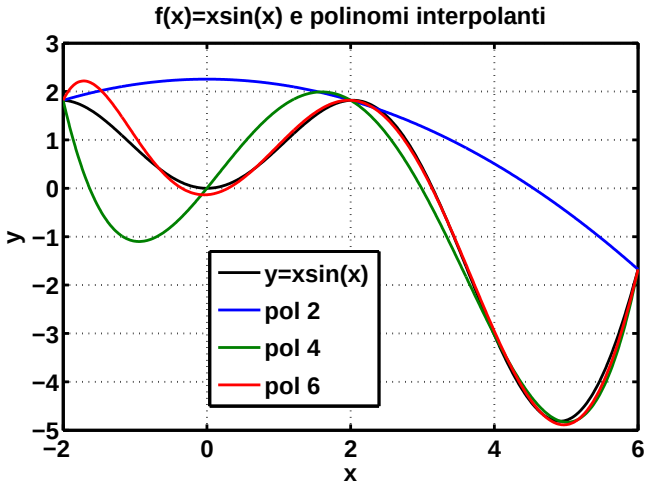

Figura 3: $f(x) = x\sin(x)$ e suoi polinomi interpolanti

È stato utilizzato il comando `plot` con parametri di ingresso un vettore, per le ascisse, ed una *matrice* `PP_dis` per le ordinate. In questo caso Matlab® interpreta ogni riga della matrice come un diverso vettore di ordinate e quindi disegnano più curve, una per ogni riga.

3. Si può rappresentare graficamente l'andamento dell'errore $\varepsilon_n(x) = |f(x) - \Pi_n(x)|$ tramite comandi analoghi ai precedenti:

```
figure( 3 )
plot( x_dis, err_dis )
grid on
legend('pol 2', 'pol 4', 'pol 6')
xlabel('x')
ylabel('y')
title('Errore di interpolazione')
```

Il grafico corrispondente è riportato in Figura 4.

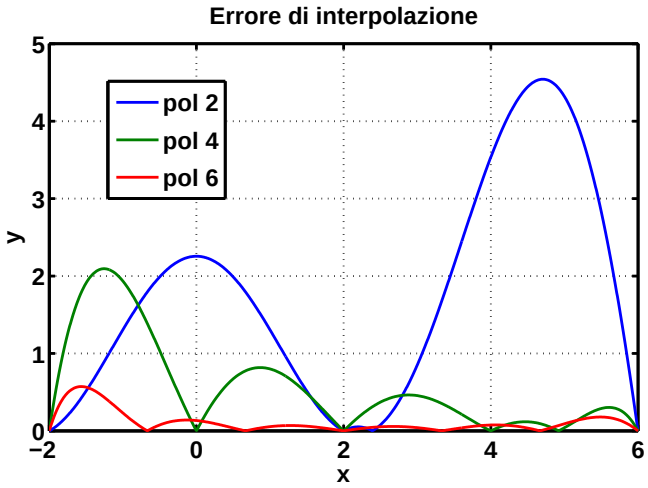

Figura 4: Errore di interpolazione.

La norma infinito

$$||\varepsilon_n(x)||_\infty = \max_{x\in[-2,6]} |f(x) - \Pi_n f(x)|$$

è già stata calcolata nell'ultima riga del ciclo `for` (vedi punto precedente) ed è pari a:

```
err_max

err_max =

    4.5425
    2.0947
    0.5724
```

rispettivamente per $n = 2,\ 4,\ 6$.

Esercizio 2

Si consideri la funzione:

$$f(x) = \sin\left(\frac{1}{1+x^2}\right) ,$$

nell'intervallo $I = [-2\pi, 2\pi]$.

1. Come nell'esercizio precedente, i polinomi di Lagrange $\Pi_n f$, interpolanti la funzione $f(x)$ nell'intervallo $[-2\pi,\ 2\pi]$ su $n+1$ nodi equispaziati ($n = 2,\ 4,\ 8,\ 10$), possono essere ottenuti tramite un ciclo `for` che iteri sui gradi come segue:

```
fun=@(x) sin((1)./(1+x.^2));
a=-2*pi;
b=2*pi;
x_dis=linspace(a,b,1000);
f_dis=fun(x_dis);
```

```
figure(1)
plot(x_dis,f_dis,'k')

err_max = [];

for n=[2 4 8 10]
x_nod=linspace(a,b,n+1);
f_nod=fun(x_nod);
P=polyfit(x_nod,f_nod,n);
P_dis = polyval( P, x_dis );
err_max = [ err_max; max( abs( P_dis - f_dis ) ) ];
hold on
plot(x_dis,P_dis)
end
title('Interpolazione f(x)', 'FontSize', 12)
legend('fun','n=2','n=4','n=8','n=10', 'Location', 'BestOutside')
```

Il codice produce i grafici riportati in Figura 5. È evidente come l'errore di interpolazione risulti maggiore ai bordi dell'intervallo rispetto ai valori assunti all'interno. Tale comportamento tende ad accentuarsi sempre di più all'aumentare del grado n, individuando il cosiddetto "fenomeno di Runge".

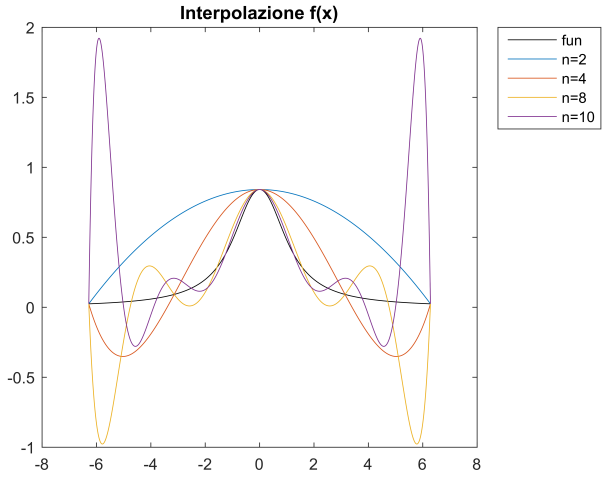

Figura 5: Grafico di $f(x)$ e $\Pi_n f$ per $n = 2, 4, 8, 10$.

2. In accordo con quanto osservato in precedenza, possiamo verificare come la norma infinito dell'errore di approssimazione cresca all'aumentare del grado polinomiale n. Si può tracciare un grafico dell'errore in funzione di n tramite le seguenti istruzioni:

```
figure(2)
plot([2 4 8 10], err_max, '-s')
title('errore in norma infinito')
xlabel('n')
ylabel('error')
```

che producono quanto riportato in Figura 6.

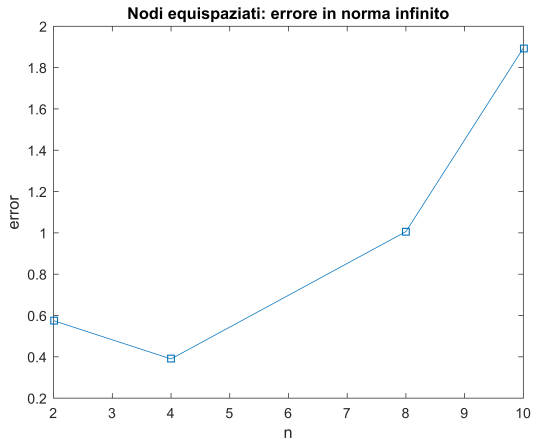

Figura 6: Norma infinito dell'errore di interpolazione in funzione di n.

3. È possibile calcolare l'errore di interpolazione della funzione $f(x)$ in norma infinito, sui nodi di Chebyshev per $n = 4, 8, 10$, mediante le istruzioni che seguono.

```
figure(3)
plot(x_dis,f_dis, 'k', 'DisplayName', 'fun');
axis([-2*pi 2*pi -0.1 .9] )
hold on
xlabel('asse x');
ylabel('asse y');
title('Interpolazione nodi Chebyshev', 'FontSize', 12);

err_max = [];
N=[2 4 8 10];
for n=N
    k=0:n;
    t=-cos(pi*k/n);           % vettore contenete i nodi di interpolazione
    x_nod=((b-a)/2)*t+(a+b)/2; % trasformazione affine
    f_nod=fun(x_nod);
    P=polyfit(x_nod,f_nod,n);
    poly_dis=polyval(P,x_dis);
    figure(3);
    h1=plot (x_dis,poly_dis, 'DisplayName', ...
        ['inter ' num2str(n) ' nodi']);
    plot (x_nod,f_nod,'o', 'Color', get(h1, 'Color'), ...
        'DisplayName', ['nodi (n = ' num2str(n) ')']);
    err_max= [err_max; max(abs(poly_dis-f_dis))];
end

figure(3)
legend( 'show','Location', 'BestOutside');

figure(4)
plot([2 4 8 10], err_max, '-s')
xlabel('n')
ylabel('errore')
title('Errore di interpolazione nodi Chebyshev', 'FontSize', 12)
```

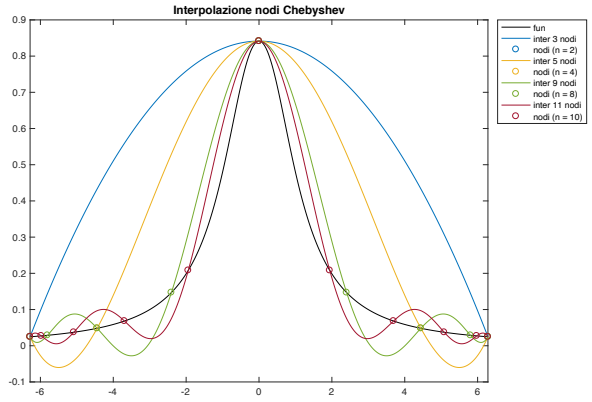

Figura 7: Interpolazione sui nodi di Chebyshev

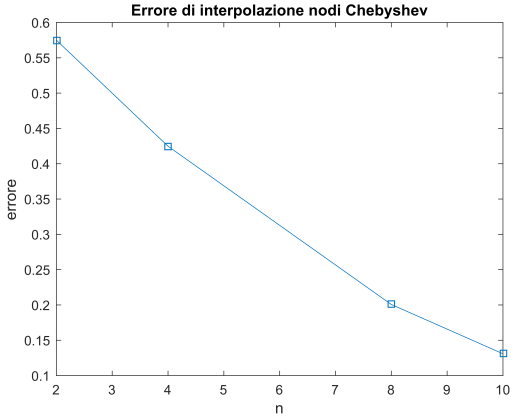

Figura 8: Errore di interpolazione sui nodi di Chebyshev

In Figura 7 sono rappresentati le interpolanti sui nodi di Chebyshev ed in Figura 8 i rispettivi errori in norma infinito in funzione del grado n. Un confronto con le Figure 5,6 permette di evidenziare immediatamente come l'utilizzo dei nodi di Chebyshev abbia ridotto drasticamente il fenomeno di Runge e quindi anche l'errore di interpolazione.

Esercizio 3

1. L'espressione analitica del polinomio interpolante di Lagrange di grado n per la funzione $f(x)$ sull'insieme di nodi distinti $\{x_0, \ldots, x_n\}$ è:

$$\Pi_n f(x) = \sum_{k=0}^{n} f(x_k)\phi_k(x),$$

dove $\phi_k(x) = \prod_{i=0, i\neq k}^{n} \frac{x-x_i}{x_k-x_i}$ sono le funzioni di base Lagrangiane associate ai nodi di interpolazione. Nel caso $n = 2$, $x_0 = 0$, $x_1 = \frac{1}{2}$ e $x_2 = 2$, tali funzioni avranno la

seguente espressione analitica:

$$\phi_0(x) = \frac{x-x_1}{x_0-x_1}\frac{x-x_2}{x_0-x_2} = x^2 - \frac{5}{2}x + 1,$$
$$\phi_1(x) = \frac{x-x_0}{x_1-x_0}\frac{x-x_2}{x_1-x_2} = -\frac{4}{3}x^2 + \frac{8}{3}x,$$
$$\phi_2(x) = \frac{x-x_0}{x_2-x_0}\frac{x-x_1}{x_2-x_1} = \frac{1}{3}x^2 - \frac{1}{6}x;$$

osservando infine che i valori di interpolazione sono $f(x_0) = -2$, $f(x_1) = -\frac{11}{8}$ e $f(x_2) = 2$, il polinomio di Lagrange di grado 2 assume la seguente espressione:

$$\Pi_2 f(x) = \frac{1}{2}x^2 + x - 2.$$

2. Ripetendo i passaggi illustrati al punto precedente, per la terna di nodi di interpolazione $x_0 = 0$, $x_1 = 1$ e $x_2 = 2$, si ottengono le seguenti espressioni per le funzioni di base Lagrangiane:

$$\phi_0(x) = \frac{1}{2}x^2 - \frac{3}{2}x + 1,$$
$$\phi_1(x) = -x^2 + 2x,$$
$$\phi_2(x) = \frac{1}{2}x^2 - \frac{1}{2}x;$$

quindi, essendo i valori di interpolazione $f(x_0) = -2$, $f(x_1) = 0$ e $f(x_2) = 2$, il polinomio di Lagrange di grado 2 assume in questo caso l'espressione $\Pi_2 f(x) = 2x - 2$, che risulta essere un polinomio di grado 1. Questo fenomeno è dovuto al fatto che le coppie di punti $\{(x_i, f(x_i))\}_{i=0}^2$ risultano allineate.

3. È sufficiente osservare che la funzione $f(x)$ è un polinomio di grado 3, per cui si ottiene $\Pi_3 f(x) = f(x)$.

Esercizio 4

Sono state svolte delle prove a trazione su una nuova lega per determinare la relazione tra lo *sforzo* σ (forza per unità di superficie, $[1000 \times \mathrm{kg_F/cm^2}]$) e la *deformazione* ε (allungamento per unità di lunghezza, [cm/cm]). I risultati delle prove sono riportati nella seguente tabella:

σ	0.1800	0.3000	0.5000	0.6000	0.7200	0.7500	0.8000	0.9000	1.0000
ε	0.0005	0.0010	0.0013	0.0015	0.0020	0.0045	0.0060	0.0070	0.0085

A partire da questi dati si vuole stimare la deformazione ε della lega in corrispondenza dei valori dei sforzo per cui non si ha a disposizione un dato sperimentale. A tal fine si utilizzano opportune tecniche di interpolazione. Gli interpolanti da utilizzare sono i seguenti:

- interpolazione polinomiale di Lagrange (`polyfit` e `polyval`);
- interpolazione polinomiale composita lineare (`interp1`);

- spline cubica naturale interpolante (`cubicspline`);
- spline cubica interpolante con condizioni di chiusura "not-a-knot" (`spline`);

si considerino inoltre le seguenti:

- approssimazioni nel senso dei minimi quadrati di grado 1, 2 e 4 (`polyfit` e `polyval`).

1. Per prima cosa si definiscono i vettori contenenti i dati sperimentali `sigma` (per lo sforzo σ) ed `epsilon` (per la deformazione ε):

```
sigma   = [0.18   0.3   0.5    0.6    0.72  0.75   0.8   0.9   1.0   ];
epsilon = [0.0005 0.001 0.0013 0.0015 0.002 0.0045 0.006 0.007 0.0085];
```

   In Figura 9 è riportato il grafico dei dati sperimentali, ottenuto tramite le seguenti istruzioni:

```
figure(1)
plot(sigma, epsilon, 'ko')
title('Dati sperimentali')
xlabel('Sforzo')
ylabel('Deformazione')
```

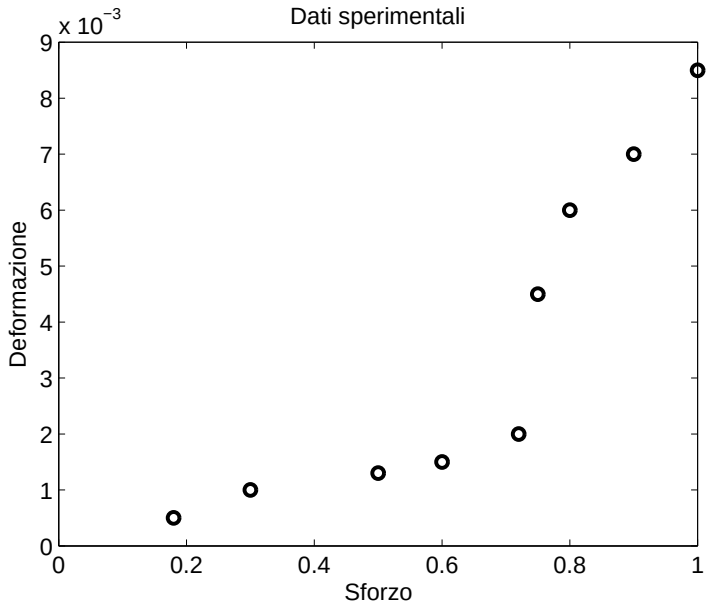

Figura 9: Dati sperimentali.

   L'*interpolazione polinomiale di Lagrange* è ottenuta mediante le funzioni `polyfit` e `polyval`. Si ricorda che il numero di punti corrispondenti ai dati sperimentali determina il grado del polinomio interpolante di Lagrange, pari al numero di punti meno uno. I coefficienti `PL` del polinomio interpolatore sono ottenuti tramite la funzione `polyfit`, mentre la valutazione del corrispondente polinomio è effettuata da `polyval`:

```
sigma_dis   = linspace(min(sigma), max(sigma), 1000);
grado       = length(sigma) - 1;
PL          = polyfit(sigma, epsilon, grado);
epsilon_IL = polyval(PL, sigma_dis);

figure(2)
plot(sigma, epsilon, 'ko', sigma_dis, epsilon_IL, 'r', 'linewidth', 2)
xlabel('Sforzo')
ylabel('Deformazione')
title('Dati sperimentali e Interp. Pol. Lagrange')
```

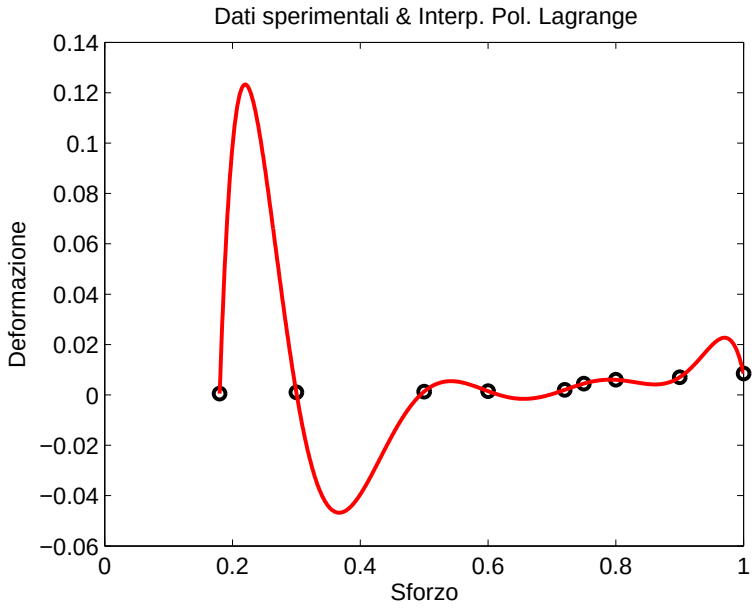

Figura 10: Dati sperimentali e interpolazione polinomiale di Lagrange.

In Figura 10 viene rappresentato il polinomio $\Pi_n(x)$ insieme ai dati. Si evince che questo tipo di interpolazione è inadeguato per la stima dei valori di deformazione lontano dai nodi, a causa del carattere oscillante del polinomio. In Matlab® è possibile ottenere l'*interpolante polinomiale composita lineare* mediante la funzione `interp1`, che prevede in ingresso tre vettori rispettivamente per le ascisse dei nodi, le ordinate dei nodi e le ascisse in cui valutare l'interpolante; in uscita sono restituite le valutazioni dell'interpolante in corripondenza delle ascisse desiderate:

```
epsilon_ICL = interp1(sigma, epsilon, sigma_dis);
figure(3)
plot(sigma, epsilon, 'ko', sigma_dis, epsilon_ICL, 'r', 'linewidth', 2)
xlabel('Sforzo')
ylabel('Deformazione')
title('Dati sperimentali e Interp. Comp. Lineare')
```

La Figura 11 mostra il polinomio interpolatore composito insieme ai dati. Si noti come l'interpolazione polinomiale composita lineare congiunga con segmenti di retta i dati sperimentali. È lo stesso tipo di interpolazione utilizzato dal comando `plot` di Matlab® per disegnare i grafici.

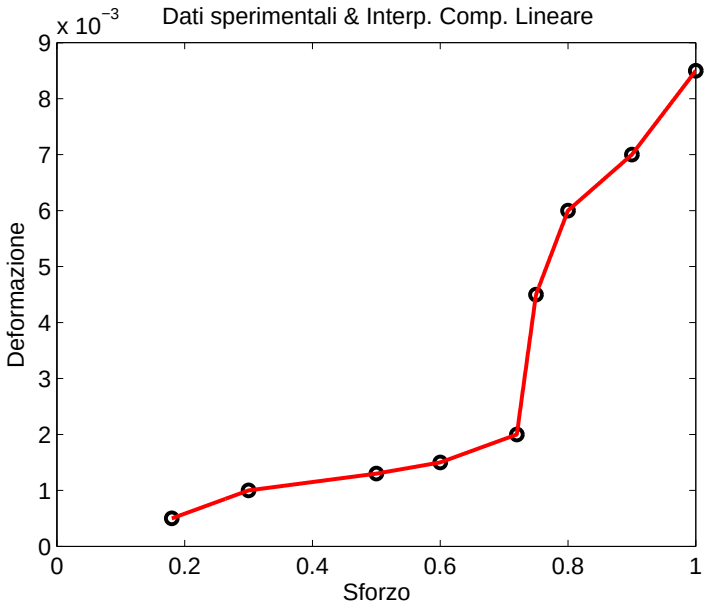

Figura 11: Dati sperimentali e interpolazione polinomiale composita lineare.

Con l'*approssimazione polinomiale nel senso dei minimi quadrati* si vogliono trovare i polinomi di grado 1, 2 e 4 di miglior approssimazione dei 9 dati sperimentali. A questo scopo si utilizza ancora la funzione `polyfit`, inserendo il grado desiderato per il polinomio ai minimi quadrati. In particolare si osservi che se `n` è un intero positivo, la chiamata di `P = polyfit(x_d, y_d, n)` corrisponde a:

- interpolazione polinomiale di Lagrange, se `n=length(x_d)-1`;
- polinomio di grado `n` nel senso dei minimi quadrati, se `n<length(x_d)-1`
- un interpolante non univocamente determinato se `n>length(x_d)-1`.

Si possono quindi approssimare le deformazioni ε con i seguenti comandi:

```
epsilon_IMQ(1,:) = polyval(polyfit(sigma, epsilon, 1), sigma_dis);
epsilon_IMQ(2,:) = polyval(polyfit(sigma, epsilon, 2), sigma_dis);
epsilon_IMQ(3,:) = polyval(polyfit(sigma, epsilon, 4), sigma_dis);

figure(4)
axes('FontSize',12)
plot(sigma, epsilon, 'ko', sigma_dis, epsilon_IMQ, 'linewidth', 2)
xlabel('Sforzo')
ylabel('Deformazione')
title('Dati sperimentali & Interp. ai Minimi Quadrati')
legend('Dati Sperimentali', ...
          'Minimi Quadrati: 1', ...
          'Minimi Quadrati: 2', ...
          'Minimi Quadrati: 4',2)
```

In Figura 12 è riportato il confronto tra le approssimazione ai minimi quadrati ed i dati sperimentali.

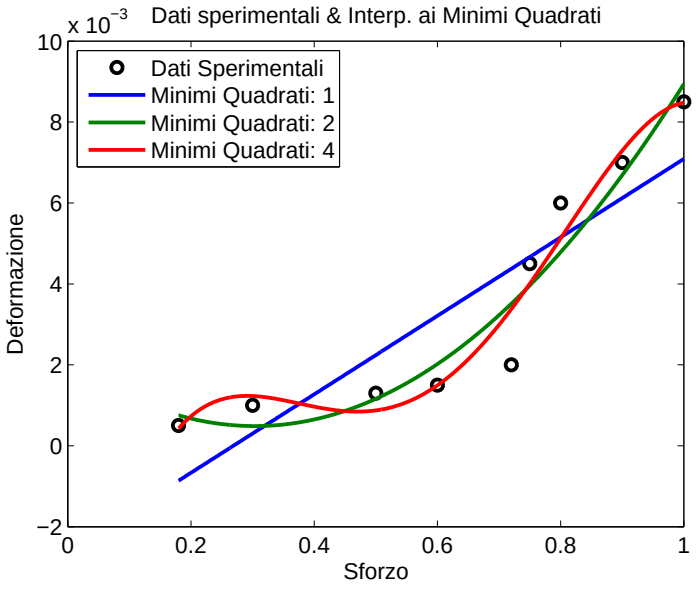

Figura 12: Dati sperimentali e approssimazione ai minimi quadrati di grado 1, 2 e 4.

L'*interpolazione mediante spline cubica naturale* può essere ottenuta tramite la funzione `cubicspline`, fornita al laboratorio. Tale funzione richiede in ingresso tre vettori rispettivamente per le ascisse dei nodi, le ordinate dei nodi e le ascisse in cui valutare la spline; in uscita viene restituita la valutazione della spline in corrispondenza delle ascisse desiderate. È perciò possibile calcolare l'approssimazione dello sforzo nei nodi richiesti tramite le istruzioni:

```
epsilon_ISCN = cubicspline(sigma, epsilon, sigma_dis);
figure(5)
axes('FontSize',12)
plot(sigma, epsilon, 'ko', sigma_dis, epsilon_ISCN, 'r', 'linewidth', 2)
xlabel('Sforzo')
ylabel('Deformazione')
title('Dati sperimentali & Spline Cubica Naturale')
```

In Figura 13 è riportato il grafico contente la spline cubica naturale. Si noti come l'approssimazione non sembri adatta in un intorno di $\sigma = 0.6 \cdot 1000 \times \mathrm{kg_F/cm^2}$; questo andamento sarà discusso al punto successivo.

L'interpolante spline cubico con condizioni "not-a-knot" può essere ottenuto con la funzione Matlab® `spline`, con la stessa sintassi di `cubicspline`. Il risultato, per cui valgono le stesse considerazioni che per la spline cubica naturale, è riportato in Figura 14.

```
epsilon_ISCnak = spline(sigma, epsilon, sigma_dis);
figure(6)
axes('FontSize',12)
plot(sigma, epsilon, 'ko', sigma_dis, epsilon_ISCN, 'r', 'linewidth', 2)
xlabel('Sforzo')
ylabel('Deformazione')
title('Dati sperimentali & Spline Cubica con condizioni not-a-knot')
```

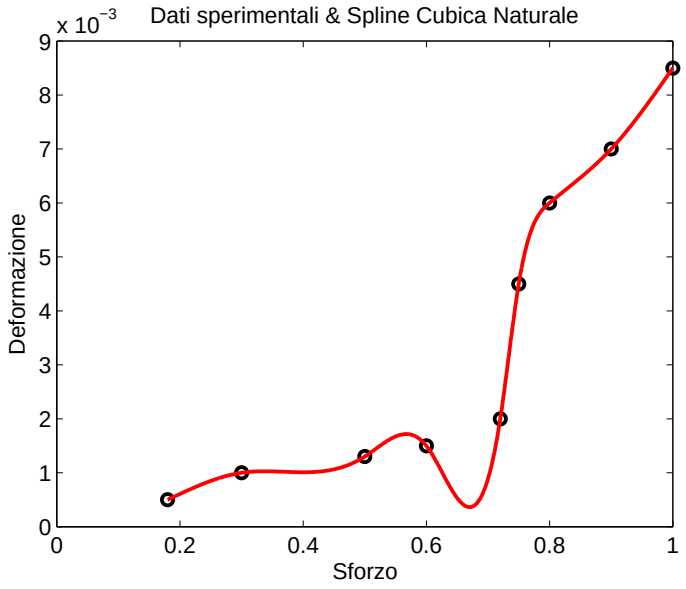

Figura 13: Dati sperimentali e spline cubica naturale.

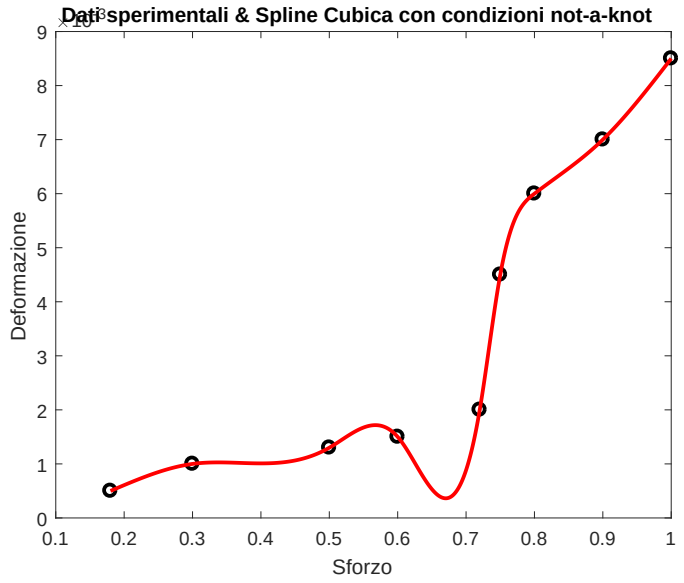

Figura 14: Dati sperimentali e spline cubica con condizioni “not-a-knot”.

2. Si vogliono ora confrontare le interpolanti e le approssimanti su un unico grafico, al fine di valutare quale approssimazione interpreti in maniera più appropriata i dati sperimentali. A questo scopo si eseguono i seguenti comandi:

```
figure(7)
axes('FontSize',12)
plot(sigma,      epsilon,          'ko', ...
sigma_dis, epsilon_IL,              ...
```

```
sigma_dis, epsilon_ICL,              ...
sigma_dis, epsilon_IMQ(3,:),         ...
sigma_dis, epsilon_ISCN,             ...
sigma_dis, epsilon_ISCnak,           'linewidth', 2)
ylim([-2e-3 1e-2])
xlabel('Sforzo')
ylabel('Deformazione')
title('Dati sperimentali & Confronti')
legend('Dati Sp.', ...
'I. Lagrange', ...
'I. Composita Lin.', ...
'Minimi Quadrati: 4', ...
'Spline Cubica N.', ...
'Spline Cubica n-a-k','Location','northwest')
```

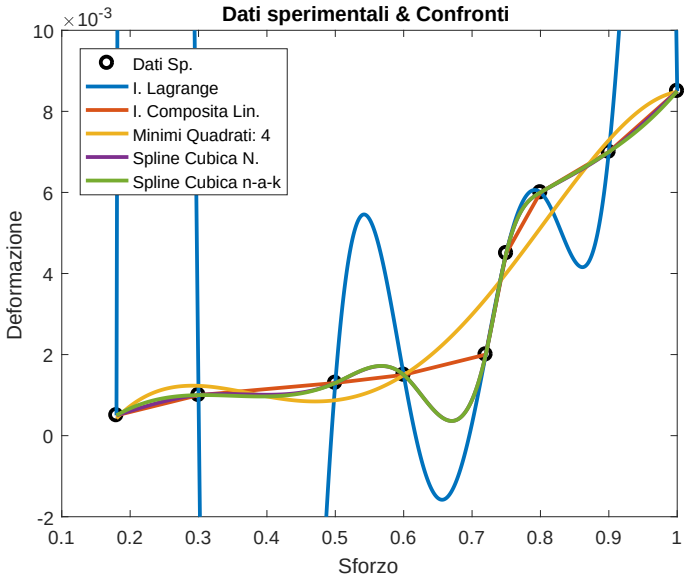

Figura 15: Confronto delle interpolanti ed approssimanti con i dati sperimentali.

In Figura 15 è riportato il confronto grafico tra tutte le interpolanti. L'interpolazione polinomiale di Lagrange è la meno adatta a descrivere il legame sforzo–deformazione della lega in sperimentazione, in quanto presenta un andamento eccessivamente oscillante. Le altre approssimazioni sembrano avere andamenti qualitativamente migliori. Nel caso di approssimazione tramite spline, la curva interpolante che descrive il legame sforzi–deformazioni ha una inversione di pendenza molto evidente attorno a $\sigma = 0.6 \cdot 1000 \times \mathrm{kg_F/cm^2}$, cioè ad un aumento dello sforzo corrisponderebbe una diminuzione della deformazione. Questo fenomeno è difficilmente giustificabile dal punto di vista fisico, per una lega di metalli, e rappresenta un limite di questa interpolazione.

3. Si possono calcolare le valutazioni della deformazione ε in corrispondenza dei valori richiesti $\sigma = 400\ \mathrm{kg_F/cm^2}$ e $\sigma = 650\ \mathrm{kg_F/cm^2}$, mediante i seguenti comandi:

```
sigma_v          = [0.4 0.65];
epsilon_IL_v     = polyval(PL, sigma_v);
```

```
epsilon_ICL_v    = interp1(sigma, epsilon, sigma_v);
epsilon_IMQ4_v   = polyval(polyfit(sigma, epsilon, 4), sigma_v);
epsilon_ISCN_v   = cubicspline(sigma, epsilon, sigma_v);
epsilon_ISCnak_v = spline(sigma, epsilon, sigma_v);

fprintf('Sigma:            %f   %f\n', sigma_v)
fprintf('Epsilon IL:       %f   %f\n', epsilon_IL_v)
fprintf('Epsilon ICL:      %f   %f\n', epsilon_ICL_v)
fprintf('Epsilon IMQ4:     %f   %f\n', epsilon_IMQ4_v)
fprintf('Epsilon ISCN:     %f   %f\n', epsilon_ISCN_v)
fprintf('Epsilon ISCnak:   %f   %f\n\n', epsilon_ISCN_v)
```

I risultati sono riassunti nella seguente tabella:

	σ	400.0 [$\mathrm{kg_F/cm^2}$]	650.0 [$\mathrm{kg_F/cm^2}$]
Lagrange	ε	-0.0396	-0.0015
Comp. Lin.	ε	0.0011	0.0017
Min. Quad. 4	ε	9.6714e-04	0.0021
Spline N.	ε	0.0010	5.3251e-04
Spline n-a-k	ε	9.6747e-04	5.3092e-04

I valori numerici confermano quanto già valutato qualitativamente in precedenza. In particolare l'interpolazione polinomiale di Lagrange risulta del tutto inadeguata per rappresentare la legge tra sforzi e deformazioni (si noti che in corrispondenza dei valori di $\sigma = 400$ $\mathrm{kg_F/cm^2}$ e $\sigma = 650$ $\mathrm{kg_F/cm^2}$ si ottengono valori negativi di deformazione). Per $\sigma = 400$ $\mathrm{kg_F/cm^2}$ l'interpolazione composita lineare, l'approssimante ai minimi quadrati di grado 4 e la spline cubica forniscono valutazioni ragionevoli; tuttavia l'interpolante composita lineare è una funzione solamente C^0 sull'intervallo considerato, mentre l'approssimante ai minimi quadrati e la spline garantiscono una maggiore regolarità (sono funzioni rispettivamente C^∞ e C^2). Le stesse considerazioni possono essere ripetute per $\sigma = 650$ $\mathrm{kg_F/cm^2}$, dove tuttavia emergono i limiti della spline che, in questo caso, fornisce un valore poco attendibile (tra $\sigma \in [600, 700]$ $\mathrm{kg_F/cm^2}$ vi è un minimo non fisico).

Esercizio 5

1. Si vuole calcolare il polinomio interpolante composito lineare $\Pi_1^H f$ su $n = 3$ sottointervalli di uguale ampiezza $H = (b-a)/n$ usando la funzione `interp1`. I seguenti comandi permettono di ricavare il grafico, riportato in Figura 16, del polinomio interpolante composito lineare:

```
fun = @(x) exp(-x.^2).*sin(x);
a = -2;
b = 3;
x_dis = linspace(a,b,1000);
f_dis = fun(x_dis);

figure(1)
plot(x_dis,f_dis,'k', 'linewidth', 2)
```

```
n=3;
h=(b-a)/n;
x_nod = a:h:b;
f_nod = fun(x_nod);
P_dis = interp1(x_nod,f_nod,x_dis);
hold on;
plot(x_nod,f_nod,'bo', 'linewidth', 2)
plot(x_dis,P_dis,'b', 'linewidth', 2)
legend('f(x)','nodi','interpolante composito lineare')
```

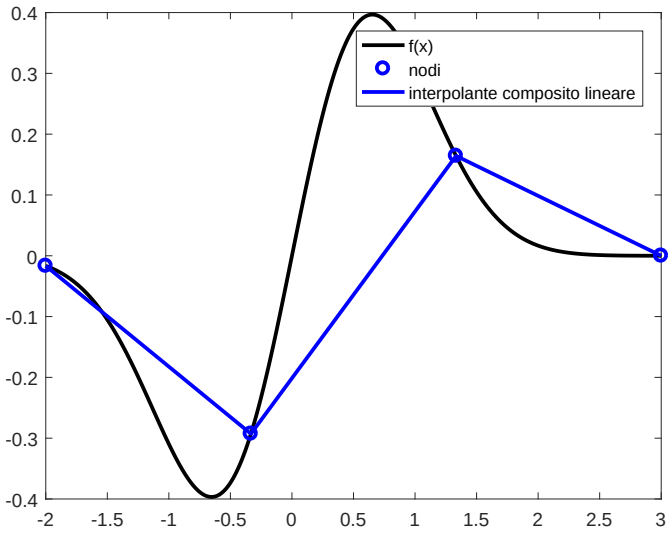

Figura 16: Grafico della funzione e del polinomio interpolante composito lineare su $n = 3$ sottointervalli.

2. L'errore di interpolazione in norma infinito

$$\epsilon_H = \max_{x\in[a,b]} |f(x) - \Pi_1^H f(x)| .$$

si ottiene valutando

```
>> err = max(abs(f_dis-P_dis))

err =
    0.4379
```

3. Per calcolare il polinomio interpolante composito lineare $\Pi_1^H f$ su $n = 4, 8, 16, 32, 64, 128$ sottointervalli di uguale ampiezza, e valutare l'errore in norma infinito ϵ_H in ciascun caso, visualizzandone l'andamento in funzione di H su un grafico in scala logaritmica su entrambi gli assi, e verificando graficamente che ci sia accordo con la stima teorica dell'errore:

$$\epsilon_H \leq \frac{H^2}{8} \max_{x\in[a,b]} |f''(x)| ,$$

è necessario inserire i comandi all'interno di un ciclo `for`:

```
H = [];
err_max=[];
for n = 2.^(2:7)
    h = (b - a)/n;
    H = [H h];
    x_nod = a:h:b;
    f_nod = fun(x_nod);
    P_dis = interp1(x_nod,f_nod,x_dis);
    err_max = [err_max;max(abs(P_dis-f_dis))];
end
```

Prima di entrare nel ciclo `for` abbiamo creato i vettori vuoti `H` e `err_max` che vengono riempiti ad ogni iterazione con i valori rispettivamente dell'ampiezza dei sottointervalli H e dell'errore ϵ_H. Alla fine del ciclo è possibile visualizzare in un grafico in scala logaritmica su entrambi gli assi l'andamento dell'errore al variare dell'ampiezza con i comandi:

```
figure(2)
loglog(H,err_max,'ro-',H,H,'k--',H,H.^2,'k', 'linewidth', 2)
legend('errore','H','H^2')
```

Il grafico è riportato in Figura 17; si verifica graficamente che l'errore ϵ_H si abbatte con andamento quadratico rispetto ad H, infatti la pendenza della curva $y(H) = \epsilon_H$ è uguale a quella della retta $y(H) = H^2$, in accordo con la stima teorica.

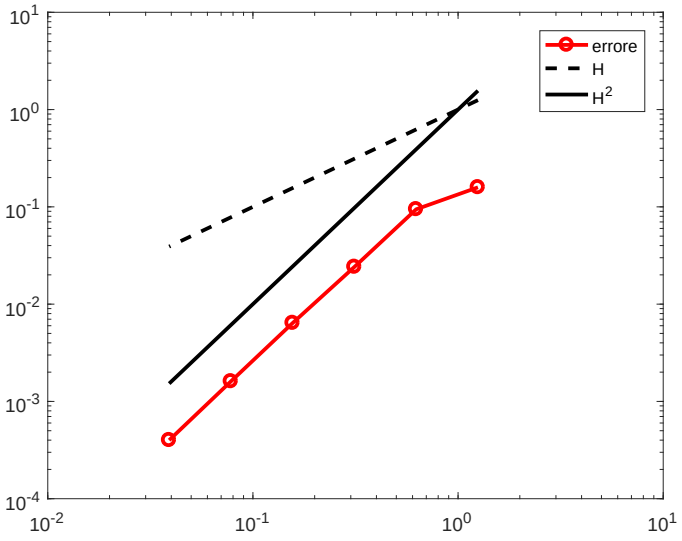

Figura 17: Errore in norma infinito ϵ_H in funzione di H e rette di riferimento con pendenza 1 e 2.

Esercizio 6

1. Si considerino i seguenti comandi Matlab® per ottenere il risultato di Figura 18:

```
T = 1;
ts = linspace( 0, T, 11 );
ys = [ 10.0 9.89 9.75 9.66 9.10 8.95 8.10 7.49 6.80 6.13 5.05 ];
y0 = 10.0;       g = 9.81;
y = @(t) y0 - 1 / 2 * g * t.^2;
tv = linspace( 0, T, 1001 );
yv = y( tv );
plot( ts, ys, 'om', tv, yv, '-b' );
grid on
xlabel( 't [s]' ); ylabel( 'y [m]' );
legend( '(ts,ys)', 'y(t)', 'Location', 'Best' );
```

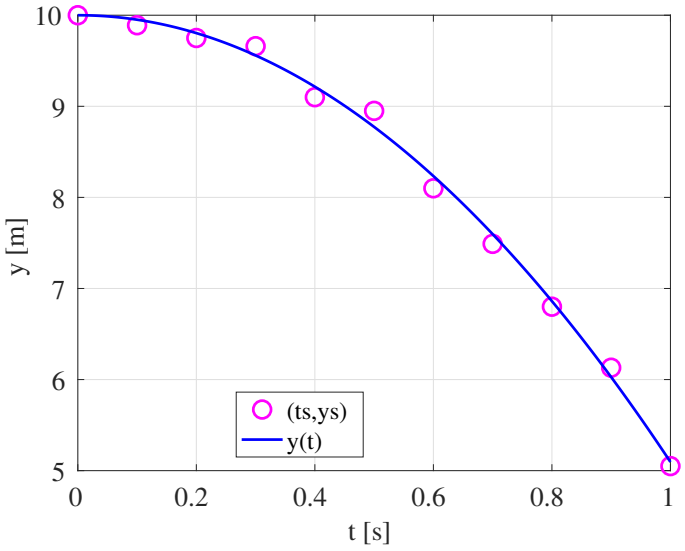

Figura 18: Dati e funzione $y(t)$.

2. Consideriamo i seguenti comandi Matlab® per rappresentare gli interpolanti e l'approssimante rappresentati in Figura 19:

```
n = length( ys ) - 1;
Pi_n_v = polyval( polyfit( ts, ys, n ), tv );  % interp. polinomiale
PiH_1_v = interp1( ts, ys, tv ); % interp. lineare a tratti
Pmq_2_v = polyval( polyfit( ts, ys, 2 ), tv );  % appross. minimi quadrati

plot( ts, ys, 'om', tv, yv, '-b', ...
      tv, Pi_n_v, '-k', ...
      tv, PiH_1_v, '-g', ...
      tv, Pmq_2_v, '-r' );
grid on
xlabel( 't [s]' ); ylabel( 'y [m]' );
legend( '(ts,ys)', 'y(t)', 'Pi_n(t)', 'PiH_1(t)', 'Pmq_2(t)', ...
        'Location', 'Best' );
```

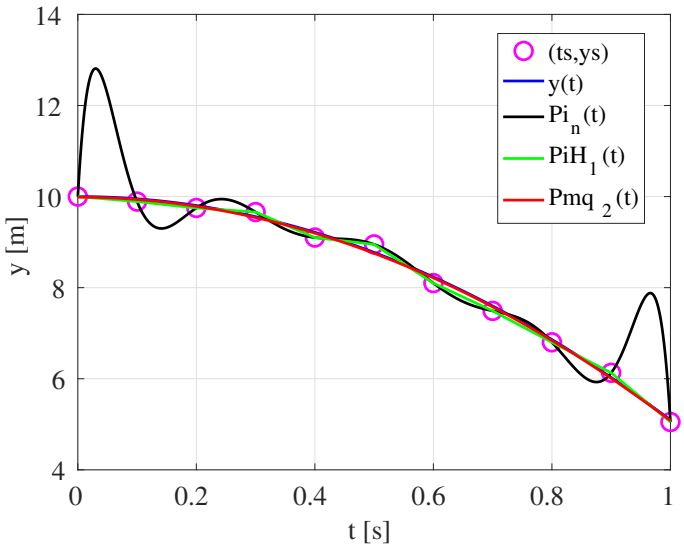

Figura 19: Dati, funzione $y(t)$, interpolante polinomiale $\Pi_n(t)$, interpolante lineare a tratti $\Pi_1^H(t)$ e approssimante polinomiale di grado $m = 2$ nel senso dei minimi quadrati $\widetilde{f}_2(t)$.

Calcoliamo ora le funzioni errore $|\Pi_n(t) - y(t)|$ per $t \in [0, 1]$, etc., che rappresentiamo in Figura 20 usando i seguenti comandi:

```
E_Pi_n_v = abs( Pi_n_v - yv );       % errori
E_PiH_1_v = abs( PiH_1_v - yv );
E_Pmq_2_v = abs( Pmq_2_v - yv );

plot( tv, E_Pi_n_v, '-k', ...
      tv, E_PiH_1_v, '-g', ...
      tv, E_Pmq_2_v, '-r'  );
grid on
xlabel( 't [s]' ); ylabel( 'errori [m]' );
legend( 'Pi_n(t)', 'PiH_1(t)', 'Pmq_2(t)', 'Location', 'Best' );
```

Osserviamo come l'interpolazione polinomiale di Lagrange non risulti adeguata a rappresentare la legge che descrive la caduta del grave sulla base dei dati sperimentali acquisiti; notiamo come tali dati siano affetti da errori di misura dello strumento. Ciò è anche confermato dal seguente calcolo degli errori $e_n = \max_{t\in[0,1]} |\Pi_n(t) - y(t)|$, etc., in cui si individuano anche i tempi in cui le funzioni errore di Figura 20 sono massime.

```
[ e_Pi_n, i_Pi_n ] = max( E_Pi_n_v );
e_Pi_n, tmax_e_Pi_n = tv( i_Pi_n )
e_Pi_n =
    2.8159
tmax_e_Pi_n =
    0.0300

[ e_PiH_1, i_PiH_1 ] = max( E_PiH_1_v );
e_PiH_1, tmax_PiH_1 = tv( i_PiH_1 )
e_PiH_1 =
    0.1762
tmax_PiH_1 =
    0.5000
```

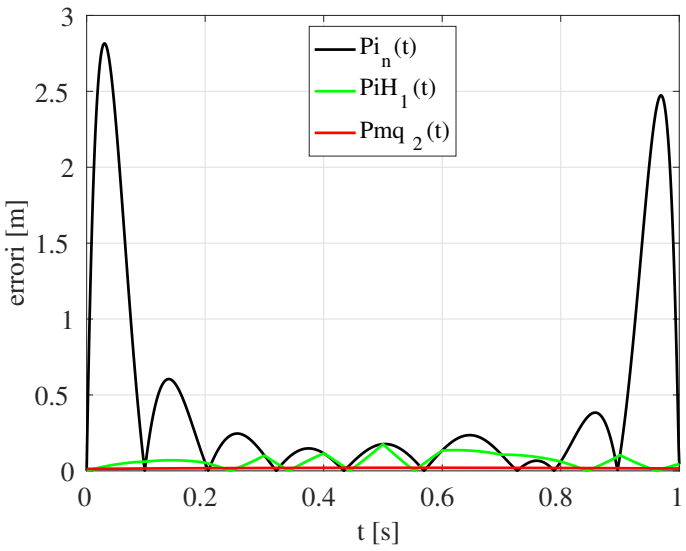

Figura 20: Errori $|\Pi_n(t) - y(t)|$, $|\Pi_1^H(t) - y(t)|$ e $|\widetilde{f}_2(t) - y(t)|$, per $t \in [0, 1]$, associati rispettivamente all'interpolante polinomiale $\Pi_n(t)$, interpolante lineare a tratti $\Pi_1^H(t)$ e approssimante polinomiale di grado $m = 2$ nel senso dei minimi quadrati $\widetilde{f}_2(t)$.

```
[ e_Pmq_2, i_Pmq_2 ] = max( E_Pmq_2_v );
e_Pmq_2, tmax_Pmq_2 = tv( i_Pmq_2 )
e_Pmq_2 =
    0.0202
tmax_Pmq_2 =
    0.5480
```

3. Consideriamo i seguenti comandi Matlab® :

```
Tf = 1.05;
tv = linspace( 0, Tf, 1001 );
Pi_n_v = polyval( polyfit( ts, ys, n ), tv );  % interpolante polinomiale
Pmq_2_v = polyval( polyfit( ts, ys, 2 ), tv );  % approssimante minimi quadrati

plot( ts, ys, 'om', tv, yv, '-b', ...
      tv, Pi_n_v, '-k', ...
      tv, Pmq_2_v, '-r'  );
grid on
xlabel( 't [s]' ); ylabel( 'y [m]' );
legend( '(ts,ys)', 'y(t)', 'Pi_n(t)', 'Pmq_2(t)', 'Location', 'Best' );
```

Si osserva dalla Figura 21 come l'interpolante polinomiale sia del tutto inadeguato ad estrapolare informazioni al di fuori dall'intervallo delle ascisse in cui i dati sono forniti; al contrario, l'approssimazione polinomiale nel senso dei minimi quadrati fornisce una stima della quota del grave molto più aderente a quanto previsto dalla legge matematica $y(t)$.

Le considerazioni precedenti sono confermate dai seguenti risultati, corrispondenti alle valutazioni di $y(t)$, $\Pi_n(t)$ e $\widetilde{f}_2(t)$ in $t = T_f = 1.05\,s$:

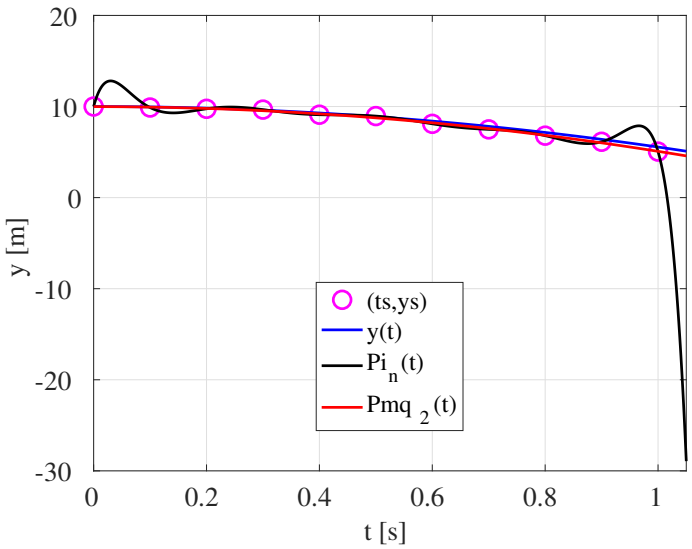

Figura 21: Dati, funzione $y(t)$, interpolante polinomiale $\Pi_n(t)$ e approssimante polinomiale di grado $m = 2$ nel senso dei minimi quadrati $\widetilde{f}_2(t)$.

```
yTf = y( Tf )
yTf =
    4.5922

yTf_Pi_n_v = Pi_n_v( end )
yTf_Pi_n_v =
  -28.9349

yTf_Pmq_2_v = Pmq_2_v( end )
yTf_Pmq_2_v =
    4.5780
```

Esercizio 7

1. Mediante i seguenti comandi Matlab è possibile ottenere il polinomio interpolante $\Pi_9 f(x)$ e l'approssimazione $\widetilde{f}_2(x)$ ai minimi quadrati, in 10 nodi equispaziati. I risultati sono riportati in Figura 22 (a sinistra).

```
a =0; b=1;
g = @(x) 10 * x.^2;
f = @(x) g(x) + 2 * rand(size(x))-1;
n = 9;
x_nodes = linspace(a,b,n+1);
y_nodes = f(x_nodes);

x_values = linspace(0,1,1001);
f_values = f(x_values);
g_values = g(x_values);
```

```
Pinterp = polyfit( x_nodes, y_nodes, n );
Pinterp_values = polyval( Pinterp, x_values );

PLeastSquares = polyfit( x_nodes, y_nodes, 2 );
PLeastSquares_values = polyval( PLeastSquares, x_values );

plot( x_values, f_values, '-g', x_values, g_values, '-k', x_values, ...
            Pinterp_values, '-r', x_values, PLeastSquares_values, '-b' )
legend( 'f(x)', 'g(x)', '\Pi_n f(x)', '\tilde f_2(x)');
```

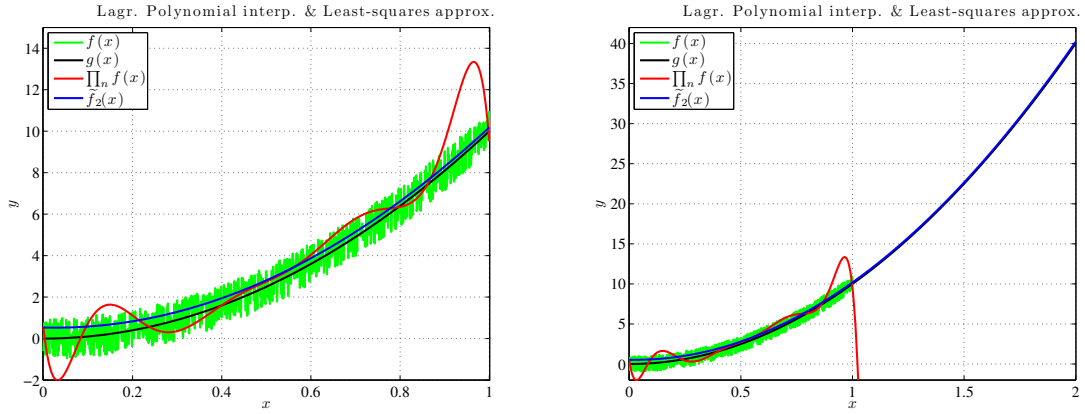

Figura 22: Polinomio $\Pi_9 f(x)$ di grado 9 interpolante f (in rosso) e parabola $\widetilde{f}_2(x)$ approssimante f ai minimi quadrati (in blu), per $x \in I = [0,1]$ (a sinistra) ed estrapolazioni su $[0,2]$ (a destra); confronto con $f(x)$ (in verde) e $g(x)$ (in nero).

Come si può osservare in Figura 22 (sinistra), l'interpolante $\Pi_9 f$ non è adeguato per la rappresentazione del segnale g nell'intervallo I, poiché interpola le coppie $\{(x_i, f(x_i))\}_{i=0}^{9}$ che sono disturbate dall'errore $\varepsilon(x_i)$. Al contrario, l'approssimazione $\widetilde{f}_2$ ai minimi quadrati descrive in maniera appropriata la funzione, nonostante siano utilizzate le stesse coppie di dati. Notiamo che $\widetilde{f}_2$ non interpola esattamente i dati, bensì rappresenta il polinomio di grado 2 che minimizza il funzionale

$$\Phi(a_0, a_1, a_2) = \sum_{i=0}^{n} \left(f(x_i) - (a_0 + a_1 x_i + a_2 x_i^2) \right)^2 ,$$

dove $\widetilde{f}_2(x) = a_0 + a_1 x + a_2 x^2$. In questo caso, otteniamo i valori $a_0 = 0.5337, a_1 = -0.5480, a_2 = 10.1978$, raccolti nella variablile `PLeastSquares`.

È opportuno sottolineare che tutti i valori ottenuti dipendono dall'output della funzione `rand`.

2. Estrapolando dai polinomi $\Pi_9 f$ e $\widetilde{f}_2$ al di fuori dell'intervallo I, mediante i seguenti comandi, si possono i grafici riportati in Figura 22 (a destra).

```
x_values2 = linspace( 0, 2, 1001 );
g_values2 = g(x_values2);
Pinterp_values2 = polyval( Pinterp, x_values2 );
```

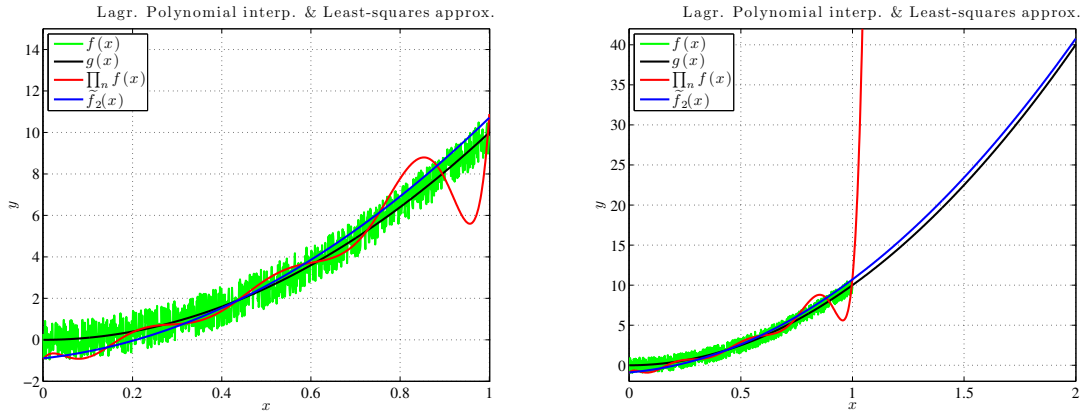

Figura 23: Polinomio $\Pi_9 f(x)$ di grado 9 interpolante f (in rosso) e parabola $\widetilde{f}_2(x)$ approssimante f ai minimi quadrati (in blu), per $x \in I = [0, 1]$ (a sinistra) ed estrapolazioni su $[0, 2]$ (a destra); confronto con $f(x)$ (in verde) e $g(x)$ (in nero) per valori del rumore $\varepsilon(x)$ diversi da quelli considerati in Figura 22.

```
PLeastSquares_values2 = polyval( PLeastSquares, x_values2 );
Pinterp_values2_x2 = Pinterp_values2( end )

Pinterp_values2_x2 =
  -3.3843e+06

PLeastSquares_values2_x2 = PLeastSquares_values2(end)

PLeastSquares_values2_x2 =
   40.2288

plot( x_values, f_values, '-g', x_values2, g_values2, '-k', ...
        x_values2, Pinterp_values2, '-r', x_values2, PLeastSquares_values2, '-b'
)
legend( 'f(x)', 'g(x)', '\Pi_n f(x)', '\tilde f_2(x)' );
```

Si osserva che l'interpolante $\Pi_9 f$ risulta ancora inadeguato a rappresentare il comportamento della funzione g e dunque fornisce un valore in $x = 2$ (pari a $\Pi_9 f(2) = -3.3843 \cdot 10^6$) che è molto distante da quello della funzione ($g(2) = 40$). L'approssimazione $\widetilde{f}_2$, invece, permette un'estrapolazione più affidabile: il valore $\widetilde{f}_2(2) = 40.2288$ è vicino al valore $g(2) = 40$, e l'errore commesso risulta nel range di variabilità $[-1, 1]$ del rumore $\varepsilon(x)$ contenuto nella funzione di misurazione $f(x)$.

3. Ripetendo i punti precedenti, la funzione `rand` restituisce dei valori differenti ad ogni chiamata: ad esempio, abbiamo ottenuto i risultati riportati in Figura 23.

   Come si può osservare, l'approssimazione ai minimi quadrati fornisce ancora una rappresentazione adeguata della funzione $g(x)$, sia nell'intervallo I, sia al di fuori, con una sensibilità molto bassa rispetto allo specifico insieme di dati $\{(x_i, f(x_i))\}_{i=0}^{9}$. Al contrario, il polinomio interpolante, oltre a non descrivere in maniera appropriata la funzione g in I e a non consentire un'opportuna estrapolazione, risulta molto sensibile alle variazioni legate al rumore ε, come si può constatare confrontando le Figure 22 e 23.