Esercitazione 1

Esercizio 1: Compressione utilizzando spline di grado 1

Dato un vettore di valori di una funzione, e.g. y=f(t) dove t=linspace(0,1,m) e m grande e una toleranza tol, si desidera estrarre un sottoinsieme di nodi $x \subset t$ con $x(1) = 0$ e $x(n) = 1$, tale che l'errore $max(abs(s - y)) < tol$ dove s è la spline di grado 1 con nodi x. Una soluzione può essere ottenuta nel seguente modo. Si comincia con il porre con $x(1) = t(1)$ il primo nodo dell'insieme. Si cicla sui punti y successivi al primo nodo e si scartano quello in cui l'errore è maggiore della tolleranza. Trovato il punto per cui questo errore nei punti interni è maggiore, diciamo $t(n)$ si inserisce nell'insieme miminale $t(n - 1)$ che è il punto per cui al passo precedente l'errore era minore della tolleranza. Successivamente si riparte considerando $t(n - 1)$ come nuovo punto iniziale.

Riportare in uscita:

1. il grafico della funzione interpolante ed interpolata;
2. il grafico dell'errore tra funzione interpolante ed interpolata;
3. il grafico del tasso di compressione al variare della tolleranza.

Una linea guida per il codice (da completare è) è la seguente:

```
% Inizializzazioni

m = % numero punti
t = linspace();
y = % funzione da interpolare mediante compressione

tol = % Tolleranza del metodo


% caso base
x  = t(1); % Salvo il punto iniziale
fx = y(1); % Salvo il suo valore associato
```

```
% numero nodi della funzioni interpolante
numnodi = length(t);

% Nodo in cui parte l'interpolazione lineare
% corrisponde al primo nodo dell'insieme x
startnode = 1;

% nodo corrente preso in considerazione
currnode = 2;

% Nodo finale di interpolazione
endnode = 3;

% ciclo dal nodo 2 fintantoche il nodo corrente non sia l'ultimo dell'interpolazione
while(endnode ≠ numnodi),
    % Calcolo il valore dell'interpolazione tra i nodi [startnode e endnode]
    % punti dell'interpolazione lineare
    xa =
    fa =
    xb =
    fb =
    % nodi dove necessaria interpolazione per confronto
    % interni all'intervallo startnode+1 a currnode
    ind = startnode+1:currnode;
    % x della funzione originaria
    xv =
    % y della funzione originaria
    fv =

    % valori calcolati mediante interpolazione
    % nei nodi interni all'intervallo preso in considerazione
    % utilizzo routine interp1
    fv_int =

    % Test di aggiornamento dell'insieme x
    % Se esiste almento un valore fuori dalla tolleranza specificata
    % bisogna aggiornare x, mi aiuto con il calcolo del massimo in valore
    % assoluto tra valori della interpolante ed interpolata.
    errore =

    % se errore maggiore della tolleranza aggiorno l'insieme x
    if( errore>tol )
        % aggiorno x e fx con valore dell'estermo di b
        x  =
        fx =
        % aggiorno start node come nodo corrente
        startnode = currnode;
    end


    % Aggiorno nodi da considerare
```

```
    currnode = currnode + 1;
    endnode = currnode + 1;
    % se ultimo nodo lo aggiungo all'insieme finale
    if(endnode == numnodi)
        x =
        fx =
    end
end

% stampo risultati a video

% 1) confronto tra funzione interpolante ed interpolata

% 2) grafico dell'errore massimo commesso

% numero punti della funzione interpolata ed interpolante
```

Risultati

Esempio $f(x) = x^2$ nell'intervallo $(0, 1)$ con $m = 101$ punti di valutazione. Tolleranza utilizzata per la compressione 0.005.

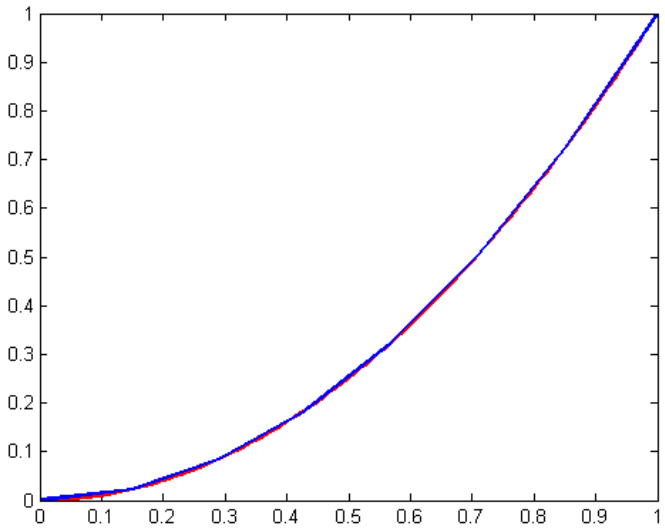

Figura 1: Risultati esercizio 1: Grafico della funzione interpolante ed interpolata.

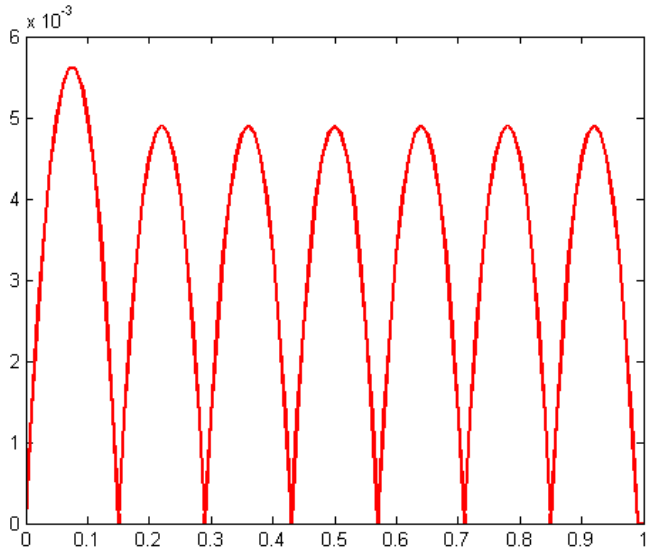

Figura 2: Risultati esercizio 1: Grafico dell'errore.

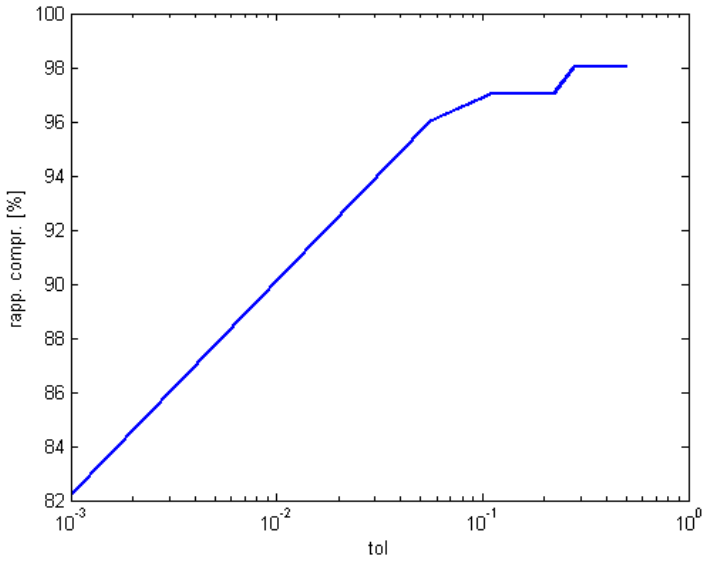

Figura 3: Risultati esercizio 1: Andamento del rapporto di compressione al variare della toleranza.

Esercizio 2: Ordine di una spline

Trovare l'ordine dell'errore uniforme fra una spline di grado 3 e una funzione f(x) liscia (ad esempio $exp(x)/cos(x)$ nell'intervallo $(0, 1)$) come una potenza di h per nodi equispaziati e condizioni di bordo del tipo:

1. naturali
2. vincolate
3. not-a-knot

Per valutare una spline naturale in MatLab si utilizza la seguente sintassi:

```
pp = csape(x,y,'variational');
v = ppval(pp,xval);
```

Per valutare una spline vincolata/complete in MatLab si utilizza la seguente sintassi:

```
pp = csape(x,y,'complete');
v = ppval(pp,xval);
```

Per valutare una spline not-a-knot in MatLab si utilizza la seguente sintassi:

```
v = spline(x,y,xval);
```

Da una suddivione dell'intervallo di interpolazione (a, b) in n suddivisioni l'ordine dell'errore uniforme può essere valutato in un numero maggiori di suddivisioni ad esempio $4n$.

Lo schema del codice può essere il seguente:

```
% Inizializzazioni

f = % funzione da interpolare
a = % estermo sinistro
b = % estremo destro


% Tipo di interpolazione
tipo = 'naturali';
% tipo = 'not-a-knot';
% tipo = 'complete';

% Inizializzazioni errore e passo h
errinf=[];
h = [];

% Ciclo al variare delle suddivisione (ad esempio da 2 a 10 in potenza di 2)
for i=2:10,
```

```
    % Numero di suddivisioni
    ndiv = 2^i;

    % punti equispaziati di interpolazione e relativa funzione
    x =
    y =

    % punti su cui valutare l'errore e relativa funzione
    xval =
    fval =

    % costruisco spline in base al tipo
    switch tipo
        case{'naturali'}

        case{'not-a-knot'}

        case{'complete'}

    end

    % valuto errore ed aggiorno relativo vettore
    errinf =
    % valuto h del passo ed aggiorno relativo vettore
    h =

end

% disegno errore in scala log-log
loglog(h,errinf)
grid on
```

Risultati

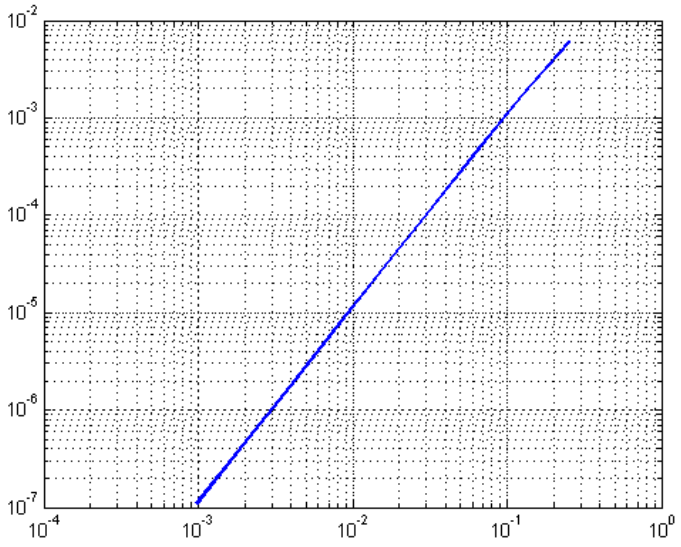

Figura 4: Risultati esercizio 2: Andamento dell'errore.

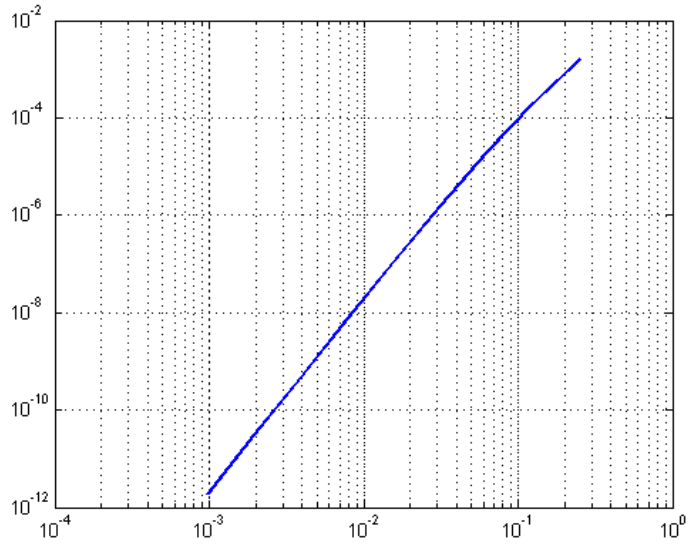

Figura 5: Risultati esercizio 2: Andamento dell'errore.

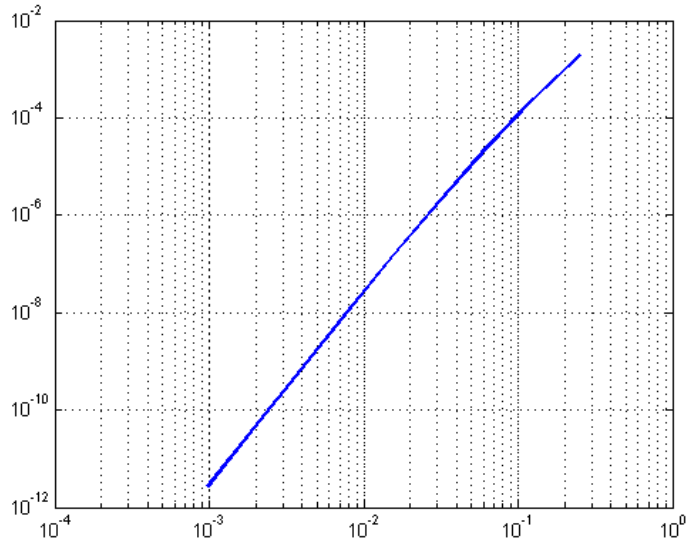

Figura 6: Risultati esercizio 2: Andamento dell'errore.

Esercizio 3: Ordine di una spline

Ripetere l'esercizio nel caso di una funzione non liscia come $\sqrt{x}$ nell'intervallo $(0, 1)$.

Risultati

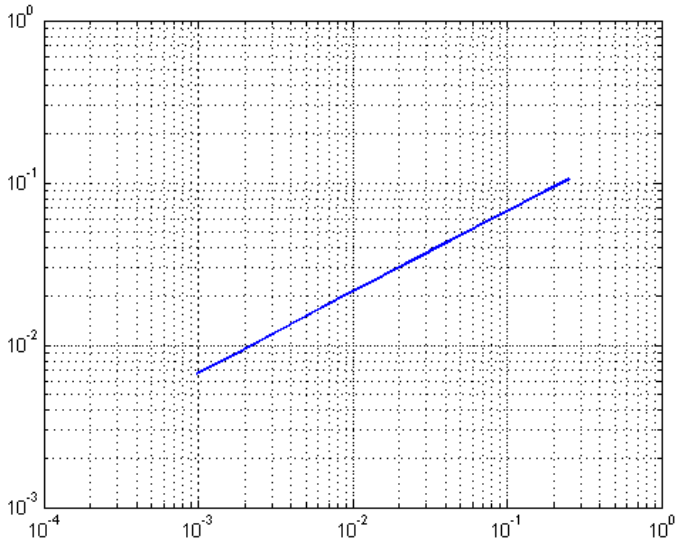

Figura 7: Risultati esercizio 3: Andamento dell'errore.

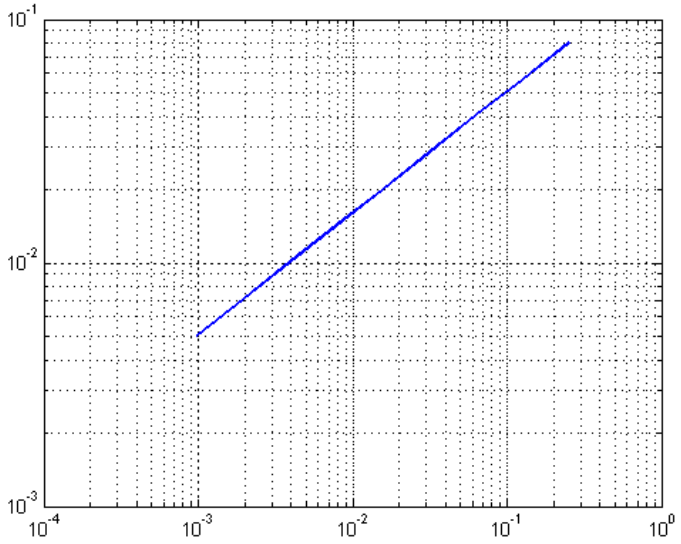

Figura 8: Risultati esercizio 3: Andamento dell'errore.

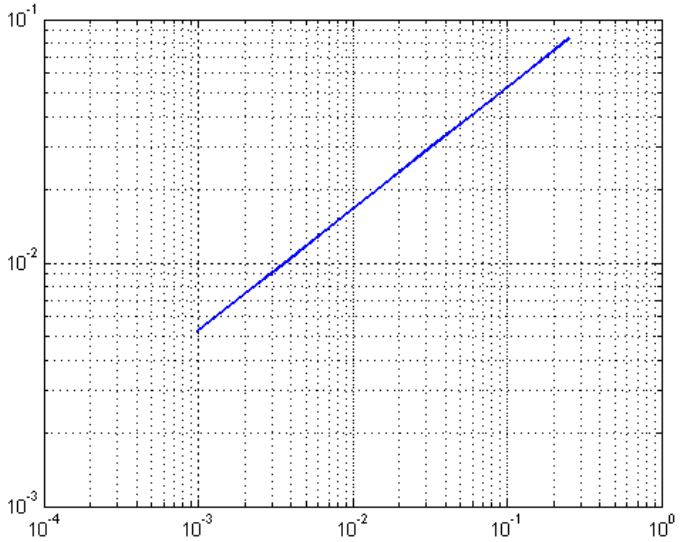

Figura 9: Risultati esercizio 3: Andamento dell'errore.

Esercizio 4: Spline quadratiche

Interpolare mediante funzioni spline quadratiche la funzione $y = atan(x)$ nell'intervallo $(0, 10)$ con 10 suddivisioni equispaziate. Cosiderare i seguenti casi:

1. spline quadratica in cui viene fissato il valore della derivata $fpa = 1$ (valore esatto di $atan$ in $x = 1$) all'estremo sinistro a.

```
% Suggerimento codice MatLab
% spline quadratica
site = [a,a,x,b,b];
xsp = [a,x];
ysp = [y(1),fpa,y(2:end)];
sp1 = spapi(site,xsp,ysp);
yintr = fnval(sp1,xval);
% derivata
sd1 = fnder(sp1);
```

2. spline quadratiche con nodi uguali ai medi dei punti di interpolazione. Questo è ottenuto attraverso il comando MatLab

```
% Suggerimento codice MatLab
% spline quadratica
sp2 = spapi(3,x,y);
yintr = fnval(sp2,xval);

% derivata
sd2 = fnder(sp2);
```

Risultati

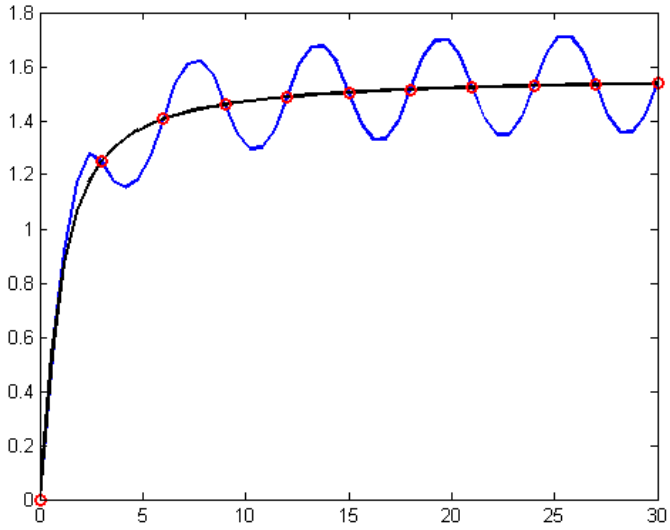

Figura 10: Risultati esercizio 4: Interpolazione quadratica caso 1.

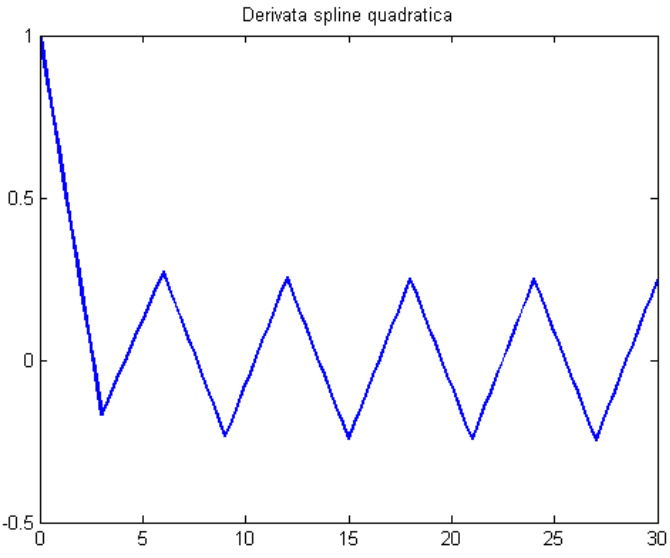

Figura 11: Risultati esercizio 4: Derivata prima caso 1.

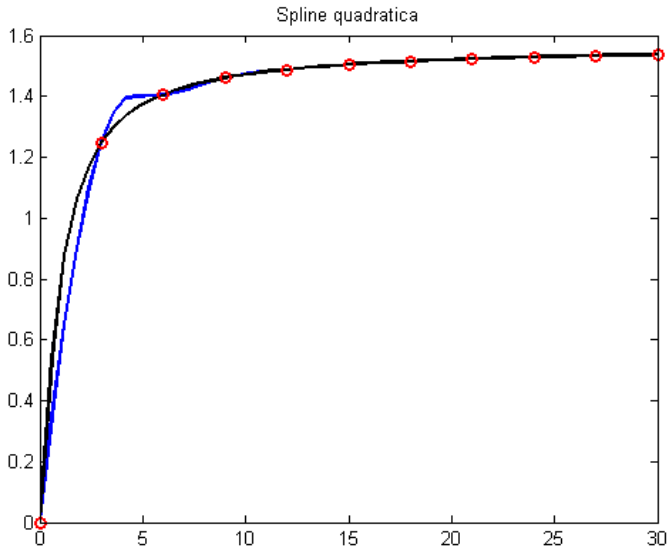

Figura 12: Risultati esercizio 4: Interpolazione quadratica caso 2.

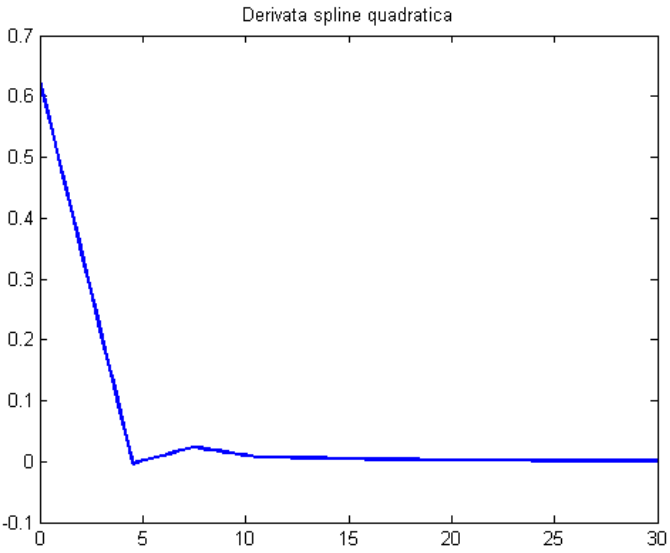

Figura 13: Risultati esercizio 4: Derivata prima caso 2.