ESERCIZI DI ANALISI NUMERICA *

A. SOMMARIVA † E M.R. RUSSO ‡

1. Alcuni problemi della teoria dell'approssimazione.

1.1. Costanti di Lebesgue. Dato un set di punti distinti $X = \{x_0, \ldots, x_n\}$ nell'intervallo chiuso e limitato $[a, b]$ si definisce *costante di Lebesgue* la quantità

$$\Lambda_n := \max_{x \in [a,b]} \sum_{i=0}^{n} |L_i(x)|$$

con al solito L_i *i-simo* polinomio di Lagrange (relativamente al set $x_0, \ldots, x_n$). Sia ora $\phi_0, \ldots, \phi_n$ una base per lo spazio $\mathbb{P}_n$ dei polinomi di grado n. Sia $V_n(X)$ la matrice di Vandermonde del set di punti X rispetto alla base $\phi_0, \ldots, \phi_n$. Sia $Y \subset [a, b]$ un sottoinsieme di discreto con cardinalità finita. Se VX e VY sono rispettivamente $V_n(X)$ e $V_n(Y)$ allora, se l'insieme Y è sufficientemente *fitto*, una buona stima della costante di Lebesgue è data da

```
leb_const=norm(VX'\VY',1);
```

D'altra parte un esempio di base è

```
function V = chebvand(deg,mymesh,intv)

% computes the Chebyshev-Vandermonde matrix at a 1d mesh
% in Chebyshev basis of a rectangle containing
% the mesh points

% INPUT:

% deg = polynomial degree;
% mymesh = 1-columns array of mesh points coordinates;
% intv = interval that contains the mesh.

%OUTPUT:

% V = Chebyshev-Vandermonde matrix

%FUNCTION BODY

if nargin < 3
    intv=[-1 1];
end

% CHEBYSHEV BASIS. SCALED.
a=intv(1); b=intv(2);
xx=(2*mymesh-a-b)/(b-a); % SCALING.
```

*Ultima revisione: 17 febbraio 2011

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

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

```
for index = 1:(deg+1)
    V(:,index)=cos((index-1)*acos(xx)); % VANDERMONDE.
end
```

Si domanda:

1. definiti per $m = n + 1$ i nodi equispaziati nell'intervallo $[-1, 1]$ che determinano l'insieme X e i nodi test

$$Y = \{-1 + hk\},\ k = 0, \ldots, 1000, h = 1/500$$

   calcolare approssimativamente la costante di Lebesgue per $n = 3, 4, 5, \ldots, 10$.
2. definiti per $m = n + 1$ i nodi di Chebyshev

$$X = \{\cos \frac{(2k+1)\pi}{2m}\},\ k = 0, \ldots, m-1$$

   e i nodi test

$$Y = \{-1 + hk\},\ k = 0, \ldots, 1000, h = 1/500$$

   calcolare approssimativamente la costante di Lebesgue per $n = 3, 4, 5, \ldots, 10$.

1.2. Miglior approssimante polinomiale e interpolante polinomiale nei nodi di Chebyshev. Il metodo di Remez permette di calcolare il polinomio di miglior approssimazione di grado n di una funzione continua f in un intervallo chiuso e limitato $[a, b]$. Esistono varie implementazioni di tale algoritmo. Utilizzando la versione descritta in

```
R. Pachon e L.N. Trefethen,
"Barycentric-Remez algorithms for best polynomial approximation
 in the chebfun system",
BIT Numer Math (2009) 49, p. 721741,
```

è possibile calcolare, fissato un grado n il polinomio di miglior approssimazione $p^* \in \mathbb{P}_1 0$.

Vediamo alcuni esempi. Si consideri la funzione $f(x) = \sin(\exp(x))$, $x \in [-1, 1]$ citata in tale pubblicazione a pagina 734. Se il sistema `chebfun` è stato installato nella versione Matlab (per il download si veda la homepage degli autori), il codice

```
% CHEBFUN DI UNA FUNZIONE.
f=chebfun('sin(exp(x))');

% POLINOMIO DI MIGLIOR APPROSSIMAZIONE p (DI GRADO 10)
% ED ERRORE COMPIUTO err.
[p,err]=remez(f,10);

% POLINOMIO p DESCRITTO NELLA BASE MONOMIALE
% pp(1)*x^N + pp(2) * x^{N-1} + ...
pp=poly(p);

% PUNTI TEST.
xx=(-1:0.001:1)';

% VALUTAZIONE DEL POLINOMIO NEI PUNTI TEST.
pxx=polyval(pp,xx);
```

```
% VALUTAZIONE DELLA FUNZIONE NEI PUNTI TEST.
fxx=feval(f,xx);

% ERRORE COMPIUTO.
err2=norm(fxx-pxx,inf);

pp'
err2
```

Si vede che l'errore ottenuto (nella norma 2 nel continuo, cioè $\int_a^b |f(x)|^2\,dx$) è approssimativamente $1.7862e-06$ e che il polinomio p_{10}^* di miglior approssimazione ha coefficienti rispetto alla base monomiale (il primo è quello di grado massimo!)

```
   0.003196437149154
   0.017324847917382
   0.033716160802325
   0.027736791074028
  -0.020560308792805
  -0.139174865843104
  -0.321930004523161
  -0.420608521955273
  -0.150685513280442
   0.540295397586175
   0.841472656602631
```

Si desidera

1. usando i comandi `polyfit` e `polyval`, valutare nei punti definiti dalla mesh $x = -1 : 0.001 : 1$ il polinomio di grado 10 che interpola la funzione

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

   nei nodi di Chebyshev

   $$\cos\frac{(2k+1)\pi}{2m},\ k = 0, \ldots, m-1.$$

   Se vogliamo calcolare il polinomio interpolante di grado 10, che valore deve assumere m?
2. calcolare una stima dell'errore in norma infinito compiuto da p nell'approssimare f, come

   $$\max_{x=-1:0.001:1} |f(x) - p(x)|$$

3. plottare il polinomio di miglior approssimazione p_{10}^* nei punti definiti dalla mesh $x = -1 : 0.001 : 1$;
4. plottare in scala semilogaritmica, nei punti della mesh $x = -1 : 0.001 : 1$, la funzione $|f - p_{10}^*|$;

2. Quadratura numerica.

2.1. Alcune formule di quadratura.

2.1.1. Esempi formule composte: trapezi e Cavalieri-Simpson. Tra i metodi comunemente utilizzati per approssimare una funzione continua f in un intervallo chiuso e limitato $[a, b]$, uno dei più semplici è la formula composta dei trapezi e di Cavalieri-Simpson. Si suddivida l'intervallo (chiuso e limitato) $[a, b]$ in N subintervalli $T_j = [x_j, x_{j+1}]$ tali che $x_j = a + jh$ con $h = (b - a)/N$. Dalle proprietà dell'integrale

$$\int_a^b f(x)\,dx = \sum_{j=0}^{N-1} \int_{x_j}^{x_{j+1}} f(x)\,dx \approx \sum_{j=0}^{N-1} S(f, x_j, x_{j+1}) \tag{2.1}$$

dove S è una delle regole di quadratura finora esposte (ad esempio $S_3(f)$). Le formule descritte in (2.1) sono dette *composte*. Due casi particolari sono

1. *formula composta dei trapezi*

$$S_1^{(c)} := h\left[\frac{f(x_0)}{2} + f(x_1) + \ldots + f(x_{N-1}) + \frac{f(x_N)}{2}\right] \tag{2.2}$$

il cui errore è

$$E_1^{(c)}(f) := I(f) - S_1^{(c)}(f) = \frac{-(b-a)}{12} h^2\, f^{(2)}(\xi),\ h = \frac{(b-a)}{N} \tag{2.3}$$

per qualche $\xi \in (a, b)$;

2. *formula composta di Cavalieri-Simpson* fissati il numero N di subintervalli e i punti $x_k = a + kh/2$ dove $h = \frac{b-a}{N}$ sia

$$I(f) \approx S_3^{(c)}(f) := \frac{h}{6}\left[f(x_0) + 2\sum_{r=1}^{N-1} f(x_{2r}) + 4\sum_{s=0}^{N-1} f(x_{2s+1}) + f(x_{2N})\right]; \tag{2.4}$$

il cui errore è

$$E_3^{(c)}(f) := I(f) - S_3^{(c)}(f) = \frac{-(b-a)}{180}\left(\frac{h}{2}\right)^4 f^{(4)}(\xi) \tag{2.5}$$

per qualche $\xi \in (a, b)$.

2.1.2. Formule gaussiane. Ricordiamo ora che una formula

$$\int_a^b f(x)w(x)dx \approx \sum_{i=1}^{M} w_i f(x_i)$$

ha grado di precisione *almeno* N se e solo se è esatta per tutti i polinomi f di grado inferiore o uguale a N. Ha inoltre grado di precisione N se e solo se è esatta per ogni polinomio di grado N ed esiste un polinomio di grado $N + 1$ per cui non lo sia. Nelle formule interpolatorie di Newton-Cotes (come ad esempio la regola del Trapezio o di Cavalieri-Simpson che sono alla base delle formule composte precedentemente citate) i nodi $x_1, \ldots, x_n$ sono equispaziati e il grado di precisione δ è generalmente uguale almeno a $n - 1$ ma in alcuni casi, come per la regola di Cavalieri-Simpson, uguale al numero di nodi n. Vediamo ora formule che a parità di nodi hanno grado di precisione maggiore di n.

Sia $w : (a, b) \to \mathbb{R}$ (non necessariamente limitato) è una funzione peso, cioè tale che

1. w è nonnegativa in (a, b);
2. w è integrabile in $[a, b]$;
3. esista e sia finito

$$\int_a^b |x|^n w(x)\, dx$$

   per ogni $n \in \mathbb{N}$;
4. se

$$\int_a^b g(x)w(x)\, dx = 0$$

   per una qualche funzione nonnegativa g allora $g \equiv 0$ in (a, b).

Tra gli esempi più noti ricordiamo

1. *Legendre*: $w(x) \equiv 1$ in $[a, b]$ limitato;
2. *Jacobi*: $w(x) = (1 - x)^\alpha (1 + x)^\beta$ in $(-1, 1)$ per $\alpha, \beta \geq -1$;
3. *Chebyshev*: $w(x) = \frac{1}{\sqrt{1-x^2}}$ in $(-1, 1)$;
4. *Laguerre*: $w(x) = \exp(-x)$ in $[0, \infty)$;
5. *Hermite*: $w(x) = \exp(-x^2)$ in $(-\infty, \infty)$;

Si dimostra che

TEOREMA 2.1. *Per ogni $n \geq 1$ esistono e sono unici dei nodi $x_1, \ldots, x_n$ e pesi $w_1, \ldots, w_n$ per cui il grado di precisione della formula di quadratura*

$$\int_a^b g(x)\, w(x)\, dx \approx \sum_{i=1}^n w_i g(x_i)$$

sia almeno $2n - 1$. I nodi sono gli zeri del polinomio ortogonale di grado n,

$$\phi_n(x) = A_n \cdot (x - x_1) \cdot \ldots \cdot (x - x_n)$$

e i corrispettivi pesi sono

$$w_i = \int_a^b L_i(x)w(x)dx = \int_a^b L_i(x)w(x)dx,\ i = 1, \ldots, n.$$

In generale il calcolo dei nodi e dei pesi non è cosa banale e rimandiamo questo dettaglio alla sezione successiva.

2.2. Implementazione delle formule di quadratura. Per quanto riguarda la formula dei trapezi, una sua implementazione in Matlab/Octave è data da

```
function [x,w]=trapezi_composta(N,a,b)

% FORMULA DEI TRAPEZI COMPOSTA.

% INPUT:
% N: NUMERO SUBINTERVALLI.
% a, b: ESTREMI DI INTEGRAZIONE.
```

```
% OUTPUT:
% x: NODI INTEGRAZIONE.
% w: PESI INTEGRAZIONE (INCLUDE IL PASSO!).

h=(b-a)/N;             % PASSO INTEGRAZIONE.
x=a:h:b; x=x';         % NODI INTEGRAZIONE.
w=ones(N+1,1);         % PESI INTEGRAZIONE.
w(1)=0.5; w(N+1)=0.5;
w=w*h;
```

che fornisce i nodi di quadratura `x` e i corrispettivi pesi `w` come due vettori colonna.

Similmente, per quanto riguarda la formula di Cavalieri-Simpson composta, i nodi di quadratura `x` e i corrispettivi pesi `w` si ottengono dalla routine

```
function [x,w]=simpson_composta(N,a,b)

% FORMULA DI SIMPSON COMPOSTA.

% INPUT:
% N: NUMERO SUBINTERVALLI.
% a, b: ESTREMI DI INTEGRAZIONE.

% OUTPUT:
% x: INTEGRAZIONE.
% w: PESI INTEGRAZIONE (INCLUDE IL PASSO!).

h=(b-a)/N;             % AMPIEZZA INTERVALLO.
x=a:(h/2):b; x=x';     % NODI INTEGRAZIONE.

w=ones(2*N+1,1);          % PESI INTEGRAZIONE.
w(3:2:2*N-1,1)=2*ones(length(3:2:2*N-1),1);
w(2:2:2*N,1)=4*ones(length(2:2:2*N),1);
w=w*h/6;
```

Osserviamo che

- in Matlab si può definire una funzione (matematica) f come *inline* e che può essere valutata con il comando *feval* (se necessario aiutarsi con l'help di Matlab/Octave), ad esempio

  ```
  >> f=inline(sin(x)'); % DEFINIZIONE COME INLINE DI UNA FUNZIONE.
  >> feval(f,0)             % VALUTAZIONE DELLA FUNZIONE.
  ans =
       0
  >>
  ```

- una volta noti il vettore (colonna) `x` dei nodi e `w` dei pesi di integrazione, se la funzione f è richiamata da un m-file *f.m*, basta

  ```
  fx=feval(f,x);                % VALUT. FUNZIONE.
  I=w'*fx;                % VALORE INTEGRALE.
  ```

  per calcolare il risultato fornito dalla formula di quadratura composta.

Per quanto concerne le formule gaussiane, esponiamo di seguito una routine eseguita da D. Laurie e W. Gautschi, che fissato un numero naturale positivo N calcola una formula gaussiana rispetto alla funzione peso di Jacobi

$$w(x) = (1-x)^{\alpha}(1+x)^{\beta},\ x \in (-1,1).$$

Osserviamo che per $\alpha = \beta = 0$, la funzione peso coincide con $w \equiv 1$.

```
function [x,w]=gauss_jacobi(N,alpha,beta)

% GAUSS-JACOBI RULE ON [-1,1] (OR (-1,1) DEPENDING ON alpha, beta).
% N: FOR GAUSSIAN RULES IT IS THE NUMBER OF QUAD. POINTS.
% alpha, beta ARE THE GAUSS-JACOBI EXPONENTS.
% x, w ARE COLUMN VECTORS OF NODES AND WEIGHTS.
%      THE LENGTH OF x AND w IS "N" IF gl=0, "N+2" IF "gl=1".

if nargin < 2
    alpha=0; beta=0;
end

ab=r_jacobi(N,alpha,beta);
xw=gauss(N,ab);

x=xw(:,1);
w=xw(:,2);

%-------------------------------------------------------------------------
% ADDITIONAL FUNCTIONS BY D.LAURIE AND W.GAUTSCHI.
%-------------------------------------------------------------------------

function ab=r_jacobi(N,a,b)

nu=(b-a)/(a+b+2);
mu=2^(a+b+1)*gamma(a+1)*gamma(b+1)/gamma(a+b+2);
if N==1
 ab=[nu mu]; return
end

N=N-1;
n=1:N;
nab=2*n+a+b;
nuadd=(b^2-a^2)*ones(1,N)./(nab.*(nab+2));
A=[nu nuadd];
n=2:N;
nab=nab(n);
B1=4*(a+1)*(b+1)/((a+b+2)^2*(a+b+3));
B=4*(n+a).*(n+b).*n.*(n+a+b)./((nab.^2).*(nab+1).*(nab-1));
abadd=[mu; B1; B'];
ab=[A' abadd];


function xw=gauss(N,ab)
N0=size(ab,1); if N0<N, error('input array ab too short'), end
J=zeros(N);
```

```
for n=1:N, J(n,n)=ab(n,1); end
for n=2:N
  J(n,n-1)=sqrt(ab(n,2));
  J(n-1,n)=J(n,n-1);
end
[V,D]=eig(J);
[D,I]=sort(diag(D));
V=V(:,I);
xw=[D ab(1,2)*V(1,:)'.^2];
```

2.3. Esempio. Vediamo di seguito un esempio su come calcolare con i codici visti un integrale di una funzione (che scegliamo regolare).

```
>> % CALCOLO SIMBOLICO IN MATLAB.
>> f=inline('1/(1+x.^2)'); % DEF. FUNZIONE.
>> syms x
>> int(1./(1+x.^2),-1,1) % INTEGRALE DEF. DA -1 A 1.

ans =

pi/2


>> % CALCOLO INTEGRALE FORMULA TRAPEZI COMPOSTA
>> [x,w]=trapezi_composta(5,-1,1);
>> length(x)

ans =

     6

>> fx=feval(f,x);
>> I_tpz=w'*fx

I_tpz =

     1.557466063348416e+00

>> err_tpz=abs(I_tpz-pi/2)

err_tpz =

     1.333026344648047e-02

>> % CALCOLO INTEGRALE FORMULA COMPOSTA SIMPSON.
>> [x,w]=simpson_composta(5,-1,1);
>> fx=feval(f,x);
>> I_simpson=w'*fx;
>> format long e
>> pi/2

ans =
```

```
     1.570796326794897e+00

>> I_simpson

     1.570795388091188e+00

>> err_simpson=abs(I_simpson-pi/2)

err_simpson =

     9.387037087638106e-07

>> length(x)  % PUNTI FORMULA COMPOSTA

ans =

    11

>> % FORMULE GAUSSIANE.
>> [x,w]=gauss_jacobi(5,0,0); % NODI [-1,1]!!!
>> fx=feval(f,x);
>> length(x)

ans =

     5

>> I_gauss=w'*fx

I_gauss =

     1.571171171171171e+00

>> err_gauss=abs(I_gauss-pi/2)

err_gauss =

     3.748443762745524e-04

>> % A PARITA' DI NODI TRAPEZI COMPOSTA OFFRIVA
>> % UN ERRORE DI 1.333026344648047e-02.
>>
>> [x,w]=gauss_jacobi(11,0,0); % NODI [-1,1]!!!
>> fx=feval(f,x);
>> I_gauss=w'*fx

I_gauss =

     1.570796336515167e+00

>> err_gauss=abs(I_gauss-pi/2)

err_gauss =
```

```
     9.720270366386785e-09

>> % A PARITA' DI NODI SIMPSON COMPOSTA OFFRIVA
>> % UN ERRORE DI 9.387037087638106e-07.
```

Si noti che in effetti il prodotto scalare `w'*fx` equivale alla valutazione della formula di quadratura

```
>> [x,w]=trapezi_composta(5,-1,1);
>> fx=feval(f,x);
>> format long e
>> I=w'*fx

I =

     1.557466063348416e+00

>> II=sum(w.*fx)

II =

     1.557466063348416e+00

>>
```

2.4. Esercizi. Quale esercizio, ricordato di scrivere le funzioni matematiche come *inline* in forma vettoriale, si approssimino (dopo averli calcolati esplicitamente o via calcolo simbolico) con le formule composte dei trapezi e di Cavalieri-Simpson come pure con le formule gaussiane (scegliere tramite α e β la funzione peso più opportuna), per $N = 5, 10, 15, 20, 25, 30$ i seguenti integrali

1. del polinomio
$$\int_{-1}^{1} x^{20}\, dx;$$
ricordando le funzioni peso di Jacobi, qual'è la funzione peso più appropriata? Il fatto che la funzione integranda sia un polinomio, spiega alcuni risultati ottenuti dalla formula gaussiana utilizzata?
2. della funzione periodica
$$\int_{-1}^{1} \sin\left(\pi\left(x+1\right)\right) dx;$$
ricordando le funzioni peso di Jacobi, qual'è la funzione peso più appropriata? (Facoltativo e richiede pazienza) Il fatto che la funzione integranda sia un polinomio, spiega i risultati ottenuti dalla formula composta dei trapezi? A tale scopo aiutarsi con il teorema di Eulero Mac-Laurin (cercare online, via motore di ricerca).
3. della funzione regolare
$$\int_{-1}^{1} \exp\left(x\right) dx;$$

ricordando le funzioni peso di Jacobi, qual'è la funzione peso più appropriata? Sono buone le stime dell'errore fornite dalle formule composte (2.3), (2.5)?

4. della funzione *holderiana* e poco regolare

$$\int_{-1}^{1} (1-x^2)^{3/2}\, dx;$$

ricordando le funzioni peso di Jacobi, qual'è la funzione peso più appropriata? Sono scadenti i risultati delle formule composte rispetto a quelle gaussiane? Nel caso delle formule gaussiane, si osservi che se $f = g \cdot w$ allora

$$\int_a^b f(x)\, dx = \int_a^b g(x)w(x)\, dx \approx \sum_i w_i g(x_i).$$

NOTA 2.2. *Per quanto concerne il calcolo simbolico, si possono utilizzare delle opportune routines presenti in Matlab per ottenere i valori esatti degli integrali. Vediamo un ulteriore esempio*

```
>> syms x
>> int((1-x.^2).^(3/2),-1,1)

ans =

(3*pi)/8

>>
```