Integrazione con Matlab

Matlab fornisce diverse funzioni built-in per l'approssimazione di integrali definiti. Vediamole assieme a degli esempi.

```
>> q=quad(FUN,a,b)
```

calcola il valore approssimato dell'integrale della funzione FUN tra gli estremi a e b, usando una formula di quadratura di Simpson adattiva. Come al solito FUN indica il nome di una function tra apici (es.: 'sin', oppure 'f' se esiste la function f.m che definisce la funzione), oppure @nomefun (nel caso precedente @f, ma anche direttamente @(x)x.*x per indicare ad esempio la funzione x^2). Attenzione però a scrivere la funzione con le notazioni vettoriali:

```
>> f=@(x)1./(x.^3-2*x+1)  % va bene
>> f=@(x)1/(x^3-2*x+1)  % non va bene
```

Il calcolo viene approssimato con una precisione per default pari a 1.e-6, ma se ne può richiedere una diversa:

```
>> q=quad(FUN,a,b,tol)
```

dove tol è appunto la tolleranza richiesta, e può esssere un solo valore oppure un vettore [tol1,tol2], con tol1=errore relativo, tol2=errore assoluto.

In alternativa possiamo usare

```
>> q=integral(FUN,a,b)
```

che ha un funzionamento analogo (si basa su di un metodo di quadratura adattiva globale), ma in questo caso FUN deve essere necessariamente del tipo @nomefun.
È in genere più veloce e accurata di *quad*, oltre al fatto che è in grado di trattare anche integrali impropri. In particolare gli estremi a e b possono ora anche essere uguali a Inf e -Inf. Per questo comando la tolleranza richiesta va indicata in modo diverso:

```
>> q=integral(FUN,a,b,'AbsTol',1.e-6,'RelTol',1e-3)
```

e il metodo si arresta quando è soddisfatta la stima dell'errore

$$| q - Int | <= \max(AbsTol, RelTol*|q|),$$

se Int indica il valore esatto dell'integrale. Per default: AbsTol=1.e-10, RelTol=1.e-6.

Esempi:

```
>> f=@(x)exp(-x.^2);
>> q=integral(f,0,Inf)  % Val_esatto=sqrt(pi)/2


>> q=integral(@(x)log(x),0,1)  % Val_esatto=-1
```

Quando non si conosce la funzione da integrare ma solo i suoi valori su di un insieme di punti, un'altra possibilità è quella di ricorrere alla formula ripetuta dei trapezi. In questo caso va fornito l'array Y dei valori su cui integrare alla funzione *trapz*:

```
>> z=trapz(Y)
```

fornisce il valore approssimato immaginando spaziatura unitaria tra i punti a cui corrispondono i valori di Y. Se la spaziatura è diversa, basta moltiplicare per il passo.

Esempio.
Immaginiamo di voler calcolare il valore dell'integrale di sin(x) tra 0 e pi greco, a partire dai valori che tale funzione assume in una partizione uniforme dell'intervallo di passo pi/100.

```
>> X=0:pi/100:pi;
>> Y=sin(X);
>> z=pi/100*trapz(Y)  % Val_esatto=2
```

(l'ultima istruzione può essere sostituita da: `>> z=trapz(X,Y)`)

Esercizio 1.
Calcolare in modo approssimato mediante le funzioni *integral* e *trapz* gli integrali di
1) sin(x)/x in (0,1); 2) (x-5)/(x^2+x+1) in (-1,1); 3) x/sqrt(x^3+1) in (0,1).

Integrali multipli

Esistono funzioni anche per l'integrazione multipla, come *integral2* (integrali doppi) e *integral3* (integrali tripli)*:*

```
>> q=integral2(FUN,xmin,xmax,ymin,ymax)
```

```
>> q=integral3(FUN,xmin,xmax,ymin,ymax,zmin,zmax)
```

Ovviamente FUN dovrà indicare rispettivamente una funzione u=f(x,y) o v=f(x,y,z), mentre xmin,xmax,ymin,ymax,zmin,zmax indicheranno gli estremi di integrazione delle rispettive variabili x,y,z. È possibile specificare come opzione aggiuntiva il metodo di integrazione da usare (v. help).

Esempio.

```
>> f2=@(x,y)x+y;
>> z=integral2(f2,0,1,0,1)  % val_esatto=1
```

Si può integrare anche su regioni non rettangolari, se alcuni estremi di integrazione sono definiti come funzioni.

Esercizio 2.
Calcolare mediante *integral2* l'integrale della funzione f(x,y)=xy nel dominio normale

$$D=\{-1 <= x <= 1,\ x^2 <= y <= 2-|x|\},$$

e controllare con i conti fatti a mano.

Grafica e integrali

Esercizio 3.
Esaminate il seguente programma che raffigura graficamente l'integrale di una funzione data mediante il metodo dei rettangoli di punto centrale, cercando di capirne il funzionamento prima di farlo girare su almeno tre esempi diversi.
Notare il modo di assegnare una funzione da input !

```
% metodo dei rettangoli con la grafica
close all
a=input('dammi l''estremo sin ');
b=input('dammi l''estremo des ');
f=input('dammi la funzione: @(x)espressione in x ');
n=input('dammi il numero di intervalli ');
x=linspace(a,b,n+1);
h=(b-a)/n;
val=integral(f,a,b);
% rettangoli
xr=a+h/2:h:b-h/2;
yr=f(xr);
rett=sum(yr)*h;
bar(xr,yr,1);
hold on
axis([a b min(min(yr),0) max(max(yr),0)])
title(['Metodo dei rettangoli, n= ',num2str(n),', valore= ',num2str(rett)]);
xlabel(['valmatlab= ',num2str(val)])
fplot(f,[a b],'r')
hold off
```

Esercizio 4.
Scrivere un programma che faccia la stessa cosa dell'esercizio precedente ma per il metodo dei trapezi.

Cenni di calcolo simbolico (non nel programma d'esame)

A volte nella risoluzione di un problema può essere utile risolvere alcuni conti in modo simbolico, come faremmo a mano, anche per non accumulare inutilmente errori di calcolo. A questo provvede il Symbolic Math Toolbox di Matlab (*se installato: attualmente non è disponibile nel Laboratorio del Centro di Calcolo, ma qualcuno potrebbe averlo sul proprio computer*). Qui non vogliamo entrare troppo nei dettagli, ma solo suggerire alcune delle cose che si possono fare. Per approfondire come al solito si consulti l'help dei comandi utilizzati negli esempi, e più in generale l'help del Symbolic Math Toolbox.

Può essere utile farlo se nei calcoli vogliamo conservare alcune costanti senza coinvolgerle nei conti (per esempio 1/3 oppure pi greco):

```
>> p=sym('pi'); third=sym('1/3'); % definizione di variabili simboliche
>> Vol=4*p*third   %volume della sfera unitaria
```

Oppure per manipolare o semplificare espressioni algebriche:

```
>> phi = sym('(1 + sqrt(5))/2');  % sezione aurea
>> f= phi^2 - phi - 1   % calcola l'espressione solo formalmente
```

```
>> simplify(f)    % semplifica l'espressione (phi è la radice del polinomio x^2-x-1 )
```

Altro esempio:

```
>> syms f g x; % definizione di variabili simboliche
>> f(x)=x^2-1; g(x)=x+1;
>> q=f/g   % espressione formale del quoziente
>> q=simplify(q)  % semplificazione del quoziente
```

Per valutare numericamente le espressioni simboliche ottenute si può usare la funzione *vpa* (variable precision arithmetic):

```
>> a=sym('1/3')
>> b=vpa(a)
>> b=vpa(a,5)   % specifica il numero di digits da usare (in questo caso 5)
```

Calcolo di integrali indefiniti

Vediamo ora qualche esempio di calcolo simbolico applicato agli integrali.

Esempio.
Si voglia calcolare l'integrale indefinito della funzione f(x)=log(x).

```
>> syms f x;
>> f(x)=log(x);
>> q=int(f,x)   % int opera l'integrazione simbolica
```

È interessante notare che in questo caso q è una normale funzione di x (perché f lo è), e quindi può essere calcolata in un valore assegnato (sempre in modo simbolico):

```
>> q(2)
        ans = 2*log(2) - 2
```

Esempio.
Calcoliamo l'integrale di una funzione razionale:

```
>> syms f x;
>> f(x)=1/(x^2+x+1);
>> g=int(f,x)    % riuscite a riconoscere la formula?
```

Esercizio 5.
Calcolate formalmente l'integrale definito tra 0 e 1 della funzione f precedente, poi valutatelo numericamente e confrontate il valore ottenuto con quello fornito dalla funzione *integral*.