Metodi Numerici

Laboratorio 4 – Quadratura numerica e test di convergenza

a.a. 2019-20

Programma di oggi

1. definire più di una function in un M-file
2. quadratura semplice
3. quadratura composita
4. test di convergenza tipo $E(N) \sim CN^{-r}$

Più di una function in un M-file

Questa tecnica è utile se per programmare una function ci fanno comodo delle function ausiliarie di cui non abbiamo bisogno altrove.

Il seguente M-file è valido:

myfunc.m

```
function F = myfunc(x,y)
  F=f(x)+g(y);
end
function b=f(a)
  b=sin(a);
end
function b=g(a)
  b=exp(a)+f(a);
end
```

Tuttavia,

- solo la function `myfunc` si può chiamare da un altro M-file o dalla riga di comando.
- le function `f` e `g` si possono chiamare solo da altre function definite in `myfunc.m`

Non è permesso mescolare `script` e `function` nello stesso M-file

Le `function` nello stesso M-file devono chiudersi con `end` (o nessuna può farlo, ma risulta meno leggibile).

Una `struct` per definire una regola di quadratura

Definiamo una regola di quadratura mediante una `struct` con i campi

- `xNodes`: array con i nodi di quadratura in $[0, 1]$
- `qWeights`: array con i pesi di quadratura in $[0, 1]$
- `gExact`: grado di esattezza polinomiale

La function `getQRule` restituisce le definizioni delle regole di quadratura del punto medio, dei trapezi e di Cavalieri-Simpson e può ovviamente essere estesa con nuove definizioni.

Quadratura semplice: `quad1.m`

Completate la function `quad1.m` che, dati in input l'integranda, gli estremi dell'intervallo e una regola di quadratura, restituisce l'integrale approssimato applicando la regola con un unico intervallo.

I polinomi fino al grado di esattezza polinomiale di ciascuna formula devono essere integrati esattamente.

Quadratura semplice: velocità di convergenza

Scegliamo una funzione con primitiva nota (e.g. e^x) e, al variare di h, calcoliamo l'integrale esatto e quello approssimato sull'intervallo $[0, h]$ così da poter calcolare l'errore esatto. Facendo un grafico log-log dell'errore al variare di h, dobbiamo osservare che:

- l'errore decresce con h
- la velocità di convergenza a zero dell'errore è legata al grado di esattezza polinomiale (sarà quindi identica per il punto medio ed i trapezi, più veloce per Cavalieri-Simpson)

Suggerimento: potete scegliere h nell'elenco generato da `HH=2.^(0:-1:-6)`, oppure `HH=logspace(0,-6,10)`

Quadratura composita: `quadN.m`

Completate la function `quadN.m` che, dati in input l'integranda, gli estremi dell'intervallo, il numero di sottointervalli, e una regola di quadratura, restituisce l'integrale approssimato applicando la regola composita.

Scelta una funzione con primitiva nota (e.g. $\sin(x)$) e un intervallo (e.g. $[0, 10]$), calcoliamo l'integrale esatto, quello approssimato con N intervalli e l'errore. Facendo un grafico log-log dell'errore al variare di N, dobbiamo osservare che:

- l'errore decresce al crescere di N
- la velocità di convergenza a zero dell'errore è legata al grado di esattezza polinomiale (sarà quindi identica per il punto medio ed i trapezi, più veloce per Cavalieri-Simpson)

Test di convergenza tipo $E(N) \sim CN^{-r}$.

Vogliamo verificare che la nostra implementazione della formula del punto medio composita abbia effettivamente un errore che decade a zero come $E(N) \sim C/N^2$ per un qualche costante $C > 0$.
Scegliamo come esempio

$$\int_0^{10} \sin(x)\mathrm{d}x = -\cos(10) + \cos(0)$$

Iniziate uno script `testQuadNconv.m` che richiami la function `quadN` sul caso scelto, con un numero di intervalli N crescente da 1 a 2^{20}. Salvate in due vettori il numero di intervalli e l'errore relativo. Realizzate un grafico in scala bilogaritmica (comando `loglog`) con il numero di intervalli in ascissa e gli errori in ordinata, mettendo un puntino per ciascun dato.

Osservate che i puntini si presentano "allineati" lungo una retta.

Teoria per il test di convergenza

Se abbiamo dei dati (x_k, y_k) "allineati" in un grafico bilogaritmico,

$$\log(y_k) \simeq M \log(x_k) + Q$$

Dimostrate che la relazione precedente è equivalente a

$$y_k \simeq e^Q (x_k)^M$$

Notare che, dati $(N, E(N))$, ponendo $C = e^Q$ e $r = -M$, la formula precedente rappresenta l'errore

$$E(N) \simeq C/N^r$$

delle formule composite.

Completate il test di convergenza

Completate il test di convergenza `testQuadNconv.m` stimando la pendenza e l'intercetta della "retta" con il metodo dei minimi quadrati (comando `polyfit`) e calcolando l'ordine di convergenza sperimentale per la formula del punto medio composito.

Dovreste ottenere $r \simeq 2$

Disegnate sul grafico bilogaritmico anche l'interpolante. Attenzione: i coefficienti della retta sono relativi alle coordinate $(\log(N), \log(E))$, ma per fare il grafico dovete tornare indietro alle coordinate (N, E)!

Dovrebbe apparire una retta che passa vicino ai dati

Attenzione ai dati che usate!

Applicate allo stesso integrale il metodo di Cavalieri-Simpson composito e ripetete il test di convergenza. L'ordine di convergenza sperimentale è quello atteso?

A cosa è dovuta la discrepanza? Correggete in modo da ottenere una stima accurata...