Calcolo numerico

Efficienza

Stabilità numerica

Sviluppo in serie di Taylor di e^x:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots, \qquad -\infty < x < \infty$$

Con resto di Lagrange

$$e^x = \sum_{k=0}^{n} \frac{x^k}{k!} = e^{\epsilon} \frac{x^{n+1}}{(n+1)!}$$

Si hanno due algoritmi per il calcolo di e^x

$$e^x \approx \sum_{k=0}^{n} \frac{x^k}{k!}$$

$$e^x \approx \frac{1}{e^{-x}} \approx \frac{1}{\sum_{k=0}^{n} \frac{(-x)^k}{k!}}$$

Il primo funziona meglio con esponente positivo, il secondo con esponente negativo

Software per calcolo numerico

- Fortran: BLAS, LAPACK
- Programmi scientifici: Matlab (licenza universitaria), Octave, Scilab
- Programmi per calcolo simbolico: Mathematica, Maple
- NumPy, SciPy
- Julia

Relazioni con altre discipline

Matematica discreta

Si cerca di utilizzare una sola formula per la risoluzione di problemi, mentre qua ci possono servire più formule

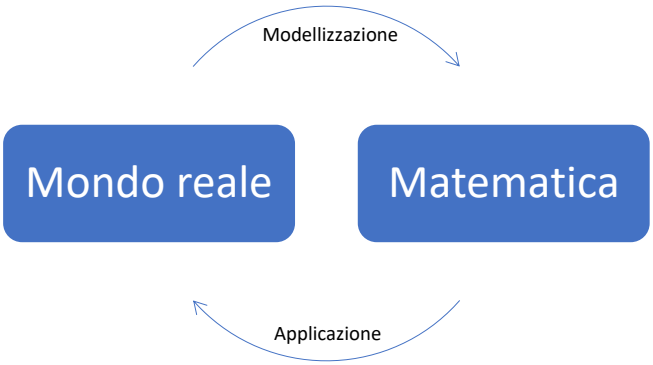
Algoritmi e strutture dati

Si occupa dello studio di problemi discreti, mentre qui ci si occupa di problemi continui

Relazione con il mondo reale

I problemi del mondo reale possono essere **modellizzati** in teorie matematiche.

Le soluzioni matematiche vengono **applicate** nel mondo reale.



Esempi di applicazioni

- Algoritmi di ricerca
- Grafica vettoriale e grafica 3D
- Multimedialità, (FFT, Fast Fourier Transform)

Error analysis

I numeri reali per la maggior parte contengono infinite informazioni, a parte i pochi numeri razionali che ne contengono di limitate.

Numeri razionali

Sono dei numeri che rispettano la descrizione di un determinato teorema, e che non rispettano altri teoremi come, ad esempio, il teorema degli zeri o quello di Lagrange.

I numeri razionali sono infiniti, noi possiamo lavorare solo con insiemi finiti di numeri.

Problema

Selezionare un numero finito di rappresentanti che possano rappresentare i numeri reali.

Distribuzione uniforme

Scegliere $\varepsilon > 0$, N pari:

$$\varepsilon = \{\varepsilon k: -\frac{N}{2} < k \leq \frac{N}{2}, k \in Z\}$$

Possiamo rappresentare i numeri reali con l'insieme

$$S = \{-\varepsilon\frac{N}{2} < x \leq \varepsilon\frac{N}{2}\} \subset N$$

Errore assoluto vs relativo

- Errore assoluto: $\tilde{x} - x$
- Erre relativo $\frac{\tilde{x}-x}{x}, x \neq 0$

Problemi con errore assoluto

La qualità dell'approssimazione con l'errore assoluto non è sempre la stessa, ad esempio per i numeri più piccoli c'è un errore più grande rispetto a quelli più grandi.

Con l'errore relativo le cifre errate nella rappresentazione dei numeri sono sempre le stesse

A noi serve sapere solo alcune cifre, quindi è meglio utilizzare una rappresentazione dei numeri con l'errore relativo migliore.

Numeri reali

Limiti di sequenze di numeri razionali che possono essere approssimate.

Data una base di numerazione $\beta \geq 2$ posso prendere un numero reale tra 0 e 1 e le sequenze di cifre

$$R \Leftrightarrow \{d_i\}_{i=1,2,3,\dots}, d_i \in \{0, \dots, \beta - 1\}$$

Teorema

Dato $x \in R\ \ 0\}$ e data una base di numerazione $\beta \geq 2$ esiste un unico $p\ \in Z$ e una sequenza $\{d_i\}_{i=1,2,3,\dots}$ Così che

1. $d_i \in \{,0,1, \dots . , \beta - 1\}$
2. $d_1 \neq 0$
3. d_i non è definitivamente uguale a $\beta - 1$ così che

$$x = \text{sign}(x)\beta^p \sum_{i=1}^{\infty} \beta^{-i} d_i$$

Il numero $\sum_{i=1}^{\infty} \beta^{-i} d_i$ si dice **mantissa**

$\pm\beta^p(0.\, d_1 d_2 d_3 \dots)$ mantissa

Floating point

Data una base di numerazione $\beta \geq 2$, il numero t > 0 di cifre della mantissa, e m, M numeri positivi, definiamo un insieme di **floating point numbers** (numeri a virgola mobile)

$$F(\beta, t, m, M) = \{0\} \cup \{\pm\beta^p \sum_{i=1}^{t} \beta^{-i} d_i\,, -m \leq p \leq M, 0 \leq d_i < \beta \text{ intero per } i = 1, \dots, t, d_1 \neq 0$$

Esempio

F := F(10, 2, 2, 3)

- $12 \in F ? \backslash implied by 12 = 0.12 \cdot 10^2$

$F \subset R$ è un insieme finito di cardinalità

$$1 + 2(m + M + 1)(\beta - 1)\beta^{t-1}$$

Il numero più grande in F è

$$\Omega = \beta^M \sum_{i=1}^{t} \beta^{-i}(\beta - 1) = \beta^M(1 - \beta^{-t})$$

Il numero più piccolo in F è

$$\omega = \beta^{-m}\beta - 1 = \beta - m - 1$$

Per $x \geq \Omega$ i numeri non possono essere rappresentati, stessa cosa per $x \leq \omega$

Esempio 2

F(2,2,1,1) è composto da 12 numeri

Questi numeri **non sono uniformi.** Tra ½ e ¼ e tra ½ e 1 esiste lo stesso numero di elementi per F.

Dato $S = \{x \in R: \omega \leq x \leq \Omega\}$ costruiamo una **funzione di rappresentazione**

$$fl: R \to F \cup \{\pm\infty\}$$

Con una delle due regole:

- Troncamento
- Arrotondamento

Se $x > \Omega$ allora il numero diventa infinito (**Overflow**)

Errore per le funzioni razionali

Errore inerente

Commentato [MV1]: Riguardare slide 148

Non valutiamo f(x) ma $f(\tilde{x})$ dove $\tilde{x} = fl(x)$

$$\varepsilon_{IN} = \frac{f(\tilde{x}) - f(x)}{f(x)}, f(x) \neq 0$$

L'errore inerente è relativamente piccolo e diciamo che il problema è **ben condizionato** o **mal condizionato**

Errore algoritmico

Non valutiamo $f(\tilde{x})\ ma\ \tilde{f}(\tilde{x})$

$$\varepsilon_{ALG} = \frac{\tilde{f}(\tilde{x}) - f(\tilde{x})}{f(\tilde{x})}, f(\tilde{x}) \neq 0$$

L'errore algoritmico è relativamente piccolo e diciamo che **l'algoritmo** f è **numericamente stabile** o **numericamente instabile**

Errore totale

$$\varepsilon_{TOT} = \frac{\tilde{f}(\tilde{x}) - f(x)}{f(x)}, f(x) \neq 0$$

Che da una misurazione genuina dell'errore nella valutazione

La situazione ideale è $|\varepsilon_{TOT}| < u$ dove u è la precisione della macchina, ma in pratica è sufficiente $|\varepsilon_{TOT}| < Mu$ con M costante.

Teorema

Sia $x \in R^n \backslash \{0\}$, e $f: R^n \to R$ razionale con $f(x) \neq 0$ e $f(\tilde{x}) \neq 0$ dove $\tilde{x} = fl(x)$ allora

$$\varepsilon_{TOT} = \varepsilon_{IN} + \varepsilon_{ALG} + \varepsilon_{IN}\varepsilon_{ALG}$$

Commentato [MV2]: Riguardare slide 158

Esempio

$$f(x) = x^2$$

Sia $\tilde{x} = fl(x) = x(1 + \varepsilon_1)$ $|\varepsilon_1| < u$ ottenuto come

$$\frac{\tilde{x} - x}{x} = \varepsilon_1 \Leftrightarrow \tilde{x} - x = x\varepsilon_1 \Leftrightarrow \tilde{x} = x(1 + \varepsilon_1)$$

$$\varepsilon_{IN} = \frac{\tilde{x}^2 - x^2}{x^2} = \frac{[x(1 + \varepsilon_1)]^2 - x^2}{x^2} = \frac{x^{2(1+\varepsilon_1)^2} - x^2}{x^2} = \frac{x^2[(1 + \varepsilon_1)^2 - 1]}{x^2} = 1 + 2\varepsilon + \varepsilon^2 - 1$$

Errore inerente

$$|\varepsilon_{IN}| = |2\varepsilon + \varepsilon^2| \leq 2|\varepsilon| + |\varepsilon^2| < 2u + u^2 = 2u, u \to 0$$

Esiste un'altra formula per calcolare l'errore inerente

$$\varepsilon_{IN} = \frac{x}{f(x)} f'(\xi)\varepsilon_x$$

Dove $\varepsilon_x = \varepsilon_1$ è la rappresentazione dell'errore in x se $f \in C^2(conv(x, \tilde{x}))$ allora

$$\varepsilon_{IN} = \frac{x}{f(x)} f'(x)\varepsilon_x + o(u)$$

Che è

$$\varepsilon_{IN} = \frac{x}{f(x)} f'(x)\varepsilon_x$$

Allora

$$\varepsilon_{IN} = \frac{x \cdot 2x}{x^2}\varepsilon_x = 2\varepsilon_x$$

Per dimostrarlo uso il teorema di lagrange applicato alla funzione utilizzando come intervallo $\tilde{x}$ –

Commentato [MV3]: Riguardare slide 163

In un intorno di 0

$$f(x) = f(0) + f'(0)x + o(x) \triangleq f(0) + f'(0)x$$

Il termine $\varepsilon_{IN} = \frac{f(\tilde{x})-f(x)}{f(x)} \doteq \frac{x}{f(x)} f'(x)\varepsilon_x$ si chiama **fattore di amplificazione** e misura l'amplificazione dell'errore.

Derivata di più variabili

Derivata di Fréchet

$$f: R^n \to R$$

$$Df(x_0): R^2 \to R$$

Se $x = (x_1, \dots, x_n), x_i \neq 0, f(x) \neq 0$

$$\varepsilon_{IN} \doteq \frac{x_1}{f(x)}\frac{\partial f}{\partial x_1}(x)\varepsilon_1 + \cdots + \frac{x_n}{f(x)}\frac{\partial f}{\partial x_n}(x)\varepsilon_n$$

Commentato [MV4]: Riguardare slide 170

Inherent error

If $x = (x_1, \dots, x_n)$, and consider $f(x)$, we have the formula (with $x_i \neq 0$, $f(x) \neq 0$)

$$\varepsilon_{IN} \doteq \frac{x_1}{f(x)}\frac{\partial f}{\partial x_1}(x)\varepsilon_1 + \frac{x_2}{f(x)}\frac{\partial f}{\partial x_2}(x)\varepsilon_2 + \cdots + \frac{x_n}{f(x)}\frac{\partial f}{\partial x_n}(x)\varepsilon_n$$

where

$$\varepsilon_i = \frac{\widetilde{x_i} - x_i}{x_i}, \qquad c_i = \frac{x_i}{f(x)}\frac{\partial f}{\partial x_i}(x),$$

are the representation error and the amplification coefficients, respectively.

Inherent error on operations

$$\varepsilon_{IN} \doteq c_1\varepsilon_1 + \cdots c_n\varepsilon_n, \qquad c_i = \frac{x}{f(x)}\frac{\partial f}{\partial x_i}(x).$$

Let $f(x,y) = xy$, the multiplication. Then $\varepsilon_{IN} \doteq c_x\varepsilon_x + c_y\varepsilon_y$, where

$$c_x = \frac{x}{f(x,y)}\frac{\partial f}{\partial x}(x,y) = \frac{x}{xy}y = 1, c_y = \frac{y}{f(x,y)}\frac{\partial f}{\partial y}(x,y) = \frac{y}{xy}x = 1.$$

Thus we have

$$c_x = 1, \qquad c_y = 1,$$

and the problem is well-conditioned. Obvious, really!

Inherent error on operations

$$\varepsilon_{IN} \doteq c_1\varepsilon_1 + \cdots c_n\varepsilon_n, \qquad c_i = \frac{x}{f(x)}\frac{\partial f}{\partial x_i}(x).$$

Let $f(x,y) = x + y$, the sum. Then $\varepsilon_{IN} \doteq c_x\varepsilon_x + c_y\varepsilon_y$, where

$$c_x = \frac{x}{f(x,y)}\frac{\partial f}{\partial x}(x,y) = \frac{x}{x+y}, \quad c_y = \frac{y}{f(x,y)}\frac{\partial f}{\partial y}(x,y) = \frac{y}{x+y}.$$

Thus we have

$$c_x = x/(x+y), \qquad c_y = y/(x+y),$$

and the problem becomes ill-conditioned when $x \approx -y$. Perhaps surprising!

Questo fenomeno si chiama **cancellazione numerica**

$$|\varepsilon_{IN}| = \left|\frac{x\varepsilon_x + y\varepsilon_y}{x+y}\right| \leq \frac{|x\varepsilon_x| + |y\varepsilon_y|}{|x+y|}$$

BLAS

Basic Linear Algebra Subroutine

Valutazione di un polinomio

Dato p e x, calcolare p(x)

$\boldsymbol{R^n}(\boldsymbol{K^n})$ vettori con n componenti in $R(K)$

$R_n(x)\big(K_n(x)\big)$ polinomi di grado al più n con coefficienti reali

$R^{nxm}(K^{nxm})$ matrici m x n con elementi in R

$R^{n_1 x n_2 x \dots x n_l}$ tensore di l dimensioni

$p(x) = a_0 + a_1 x + \cdots + a_n x^n$

In computer grafica quasi tutto dipende da delle curve, le superfici possono essere curve, ecc…

Il calcolatore per disegnare il grafico di una funzione conta numerosi puntini all'interno del grafico della funzione e ne valuta il valore.

$$\sum_{i=0}^{n} a_i x^i$$

s=0

for i=0 : n

  s=s + a(i) * x^i

end

ho $\frac{(n+1)(n+2)}{2}$ operazioni

s=0

p=1

for i = 0 : n

  s= s + a(i) * p

  p = p * x

end

il numero totale è 3 (n+1) operazioni

Algoritmo di Ruffini-Horner

$$a_0 + a_1 x + a_2 x^2 + a_3 x^3$$

$$\big((a_3 x + a_2)x + a_1\big)x + a_0$$

$$\big(\big(\dots\big((a_n x + a_{n-1})x + a_{n-2}x + \cdots\big)x + a_1\big)x + a_0$$

$$s_n = a_n si = si + 1x + ai$$

$\begin{cases} s_n = a_n \\ s_i = s_{i+1}x + a_i \end{cases}$ metodo di Ruffini-Horner

S=a(n)

For i = n-1 : -1 : 0

  S= s * x + a(i)

End

2n operazioni

Vettori

$$w, v \in K^n$$

$$v =< v_1, \dots, v_n >$$

$$w =< w_1, \dots, w_n >$$

$$v + w =< v_1 + w_1, \dots, v_n + w_n >$$

Prodotto riga per colonna tra due vettori

$$v^t w =< v_1, \dots, v_n >< w_1, \dots, w_n >= v_1 w_1 + v_2 w_2 + \cdots + v_n w_n$$

Prodotto matrice per vettore

$$A \in K^{mxn}$$

$$b \in K^n$$

$$c = Ab \in K^m$$

$$A =< r_i^t, \dots, r_m^t >$$

Faccio il prodotto riga per colonna tra le righe della matrice e il vettore.

Ma ci sono anche altre interpretazioni, come quello impiegato nel deep learning

Prodotto matrice per matrice

Stessa cosa di discreta

Prodotto tra matrici di Strassen

Algoritmo per semplificare il prodotto tra due matrici 2x2.

La moltiplicazione riga per colonna impiega 8 moltiplicazioni e 4 addizioni.

L'algoritmo di strassen impiega 7 moltiplicazioni e 18 addizioni.

Può sembrare che impieghi un tempo peggiore rispetto alla riga per colonna, ma la moltiplicazione tra due vettori impiega molto di più che l'addizione, rendendo quindi l'algoritmo più veloce.

L'algoritmo viene poi trasformato in moltiplicazioni in blocchi. Rendendo qualsiasi moltiplicazione tra matrici 2^n x 2^n come 7 moltiplicazioni e alcune addizioni tra 4 matrici 2^(n-1) x 2^(n-1) ricorsivamente.

Questo ha fatto scaturire una corsa all'algoritmo più veloce per la risoluzione di prodotti tra matrici.

Ma questi algoritmi non vengono usati.

Somme di elementi in parallelo

Ho un vettore di 2^n elementi che devono essere sommati tra loro.

Potrei scorrere tutto il vettore e sommare il primo elemento con il secondo, poi con il terzo ecc…

Ma così sarebbe troppo lento.

Posso quindi sommare in parallelo gli elementi a due a due, creando così un secondo vettore di 2^(n-1) elementi.

Ripetendo l'operazione finché non arrivo a un vettore di 1 elemento (2^1) ottengo lo stesso risultato, soltanto che ho eseguito le operazioni parallelamente, e quindi con vettori molto grandi sarebbe molto più veloce.

Valutazione di un polinomio

S=a_0, p=1

For i = 1 : n

  P= p*x

  S=s+a_i * p

3n operazioni

Ruffini-Horner

S=a_n

For i = n-1 : -1 : 0

  S= s * x + a_i

2n operazioni

Il metodo di Ruffini-Horner può essere interpretato come moltiplicazioni di matrici, che quindi può essere eseguito in parallelo, che impiega 2 log_2 (n) cicli macchina

Matrice sparsa

Una matrice che ha molti 0, molti elementi nulli

$$A \in K^{nxn}$$

$$nZ(A) \ll n^2$$

<< molto minore

Si riesce ad implementare il prodotto matrice-vettore con 2l operazioni anziché $2n^2$ operazioni

$$for(i,j) \in \Omega$$

$$c(i) += a(i,j) \cdot b(j)$$

$$\begin{cases} mat\ i \\ mat\ j \\ mat\ value \end{cases}$$

Mat[h] è una terna (o quaterna se una lista)

$$for\ h = 1{:}\,l$$

$$c(mat[h].i) += mat[h].value \cdot mat[h].j$$

Altre strutture

- Matrici sparse
- Sottospazi vettoriali
- Matrici diagonali (prodotto matrice-vettore, prodotto righe per colonne)
  - $\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$
  - $D_n = \{A \in K^{nxn}, a_{ij} = 0, i \neq j\}$
  - Le matrici diagonali sono sparse?
    - $nZ(A) \leq n$
    - $\lim_{n\to\infty} \frac{nnZ(A)}{n^2} = 0$
    - Dono strutture sparse, sono sufficienti 2l operazioni
  - È un sottospazio vettoriale?
    - $A, B \in D_n, \alpha \in K \Rightarrow A + B \in D_n, \alpha A \in D_n$
    - Per dimostrare $C = A + B \in D_n$ è sufficiente dimostrare che $c_{ij} = 0, i \neq j$
    - $c_{ij} = a_{ij} + b_{ij} = 0 + 0 = 0, i \neq j$
    - Stessa cosa per $C = \alpha A$
    - $c_{ij} = \alpha a_{ij} = \alpha 0 = 0, i \neq j$
    - È una sottoalgebra di K^{nxn}?
    - Un'algebra è un sottospazio vettoriale in cui è definito un prodotto compatibile con le operazioni di spazio vettoriale e rispetto il prodotto V è un gruppo
      - $A, B, C \in V,\ \alpha \in K$
      - $A(B + C) = AB + AC$
      - $(A + B)C = AC + BC$
      - $\alpha(AB) = (\alpha A)B = A(\alpha B)$
    - Matrici nxn K^{nxn} è un'algebra con il prodotto riga per colonna
    - Polinomi $K[x], p, q, (p + q)\tilde{p} = p\tilde{p} + q\tilde{p}$
    - Le matrici diagonali D_n sono una sottoalgebra di K^{nxn}?
      - $A, B \in D_n \Rightarrow AB \in D_n$
      - $C \coloneqq AB$
      - $c_{ij} = \sum_{l=1}^{n} a_{il} b_{lj} = a_{ii} b_{ij}$
      - $a_{il} = 0, i \neq l$
      - $b_{ij} = 0, i \neq j$

Commentato [MV5]: Riguardare lezione 14/11

Algoritmi per prodotto tra matrici diagonali

Algoritmo non strutturato

```
For i = 1:n
    For j=1:n
        C(i, j) = 0
        For l=1:n
            C(i,j) = c(i,j) + a(i, l) * b(l, j)
```

L'algoritmo strutturato si può ottenere spesso partendo dal codice di quello non strutturato

Algoritmo strutturato

C = 0

For i=1:n

C(i,i)=a(i,i) * b(i,i)

Prodotto per un vettore

Prodotto di una matrice diagonale per un vettore

For i=1:n

C(i) = a(i,i) * b(i)

Strutture e algoritmi strutturati

Strutture: sottospazi vettoriali, sottoalgebre

Algoritmo strutturato: algoritmo che sfrutta la struttura

Esempio

- Matrici diagonali: sottoalgebra
- Matrici triangolari: sottoalgebra
- Matrici tridiagonali: sottospazio vettoriale $tridiagonale = \{a_{ij} = 0, |i-j| > 1\}$

Prodotto colonna per riga

$$\{A \in K^{nxn} : A = uv^T, u, v \in K\} = D$$

$$\begin{bmatrix} u_1 \\ \vdots \\ u_n \end{bmatrix} [v_1, \dots, v_n] = \begin{bmatrix} u_1v_1 & \cdots & u_1v_n \\ \vdots & \ddots & \vdots \\ u_nv_1 & \cdots & u_nv_n \end{bmatrix}$$

È una sottoalgebra di K^{nxn}?

1. $A, B \in D \Rightarrow A + B \in D$
2. $\alpha \in K, A \in D \Rightarrow \alpha A \in D$
3. $A, B \in D \Rightarrow AB \in D$

3. $A = uv^T, B = \tilde{u}\tilde{v^T}$
   a. $AB = (uv^T)(\tilde{u}\tilde{v^T}) = u(v^T\tilde{u})\tilde{v}^T = [(v^T\tilde{u})u]\tilde{v}^T$
2. $A = uv^T$
1. $A = uv^T, B = \tilde{u}\tilde{v}^T$
   a. $A + B = uv^T + \tilde{u}\tilde{v}^T = uv^T + u$

Commentato [MV6]: Riguarda 14/11 sta machinetta

$$D = \{A \in K^{nxn} : aK(A) \leq 1\} = \{A \in K^{nxn} : A = uv^T, u, v \in K\}$$

$$A \in D, w \in K^n, Aw \sim 2n^2 ops$$

Si può trovare un algoritmo che calcola il prodotto Aw in O(n) operazioni

$$A = uv^T$$

$$Aw = (uv^T)w = u(v^Tw)$$