Calcolare funzioni di matrici

Note Title 2024-03-25

Se A diagonalizzabile, prima strategia:

$$f(A) = f(V \Lambda V^{-1}) = V \begin{bmatrix} f(\lambda_1) & & \\ & f(\lambda_2) & \\ & & \ddots \\ & & & f(\lambda_n) \end{bmatrix} V^{-1}$$

Funziona bene se A normale (ciò che possiamo scegliere V ortogonale)

Anche se calcoliamo $f(\lambda_i)$ con errore $\left| \tilde{d}_i - f(\lambda_i) \right| \leq \varepsilon$

$$\left\| V \tilde{D} V^{-1} - f(A) \right\| = \left\| V \begin{bmatrix} \tilde{d}_1 - f(\lambda_1) & & \\ & \ddots & \\ & & \tilde{d}_n - f(\lambda_n) \end{bmatrix} V^{-1} \right\|$$

$$\leq \|V\| \cdot \varepsilon \cdot \|V^{-1}\| \leq \varepsilon \cdot k(V)$$

Esempio su Matlab: $f\left(\begin{bmatrix} 3 & -1 \\ 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 10^{-5} \end{bmatrix} \right)$, diagonalizzando

$f(x) = \sqrt{x}$

Alternativa: polinomi di interpolazione

Prendiamo per esempio $p(x)$ tale che $p(2) = f(2) = \sqrt{2}$, $p'(2) = f'(2) = \frac{1}{2}\frac{1}{\sqrt{2}}$

$$p(x) = \frac{1}{\sqrt{8}} x + \frac{1}{\sqrt{2}}$$

$p(A)^2 - A \approx 10^{-15}$ in Matlab.

$p(A) \approx f(A)$ non è un'uguaglianza esatta!

$p(x)$ non è il polinomio di interpolazione in $\Lambda(A)$, ma quello in $\Lambda\left(\begin{bmatrix} 3 & -1 \\ 1 & 1 \end{bmatrix} \right)$. Però fornisce una buona approssimazione

Più giustificato: sviluppo di Taylor.

$$f(x) = f(\alpha) + f'(\alpha)(x-\alpha) + \frac{f''(\alpha)}{2}(x-\alpha)^2 + \dots$$

$$f(A) = f(\alpha)I + f'(\alpha)\left(A-\alpha I\right) + \frac{f''(\alpha)}{2}\left(A-\alpha I\right)^2 + \dots$$

In questo caso, è naturale prendere $\alpha = 2$. In generale, una buona scelta di α per una matrice generica è

$\alpha = \frac{\lambda_1 + \lambda_2 + \dots + \lambda_n}{n} = \frac{1}{n}\mathrm{Tr}(A)$, facile da calcolare senza eig(..).

Esempio del comportamento opposto:

$$A = \begin{bmatrix} 0 & 30 \\ -30 & 0 \end{bmatrix} \qquad \exp(A) = \begin{bmatrix} \cos(30) & \sin(30) \\ -\sin(30) & \cos(30) \end{bmatrix}$$

Termini dell'ordine di 10^{11} nella serie di Taylor

+ cancellazione per produrre risultato $\approx 10^{-1}$

$\Rightarrow$ errori dell'ordine di 10^{-5}

Abbiamo scelto lo stesso procedimento: $\alpha = \frac{1}{n}\mathrm{Tr}(A) = 0$, serie di Taylor in α.

Gli autovalori però sono distanti da $\alpha \Rightarrow$ convergenza lenta.

Questa crescita intermedia è un fenomeno comune, specialmente se A non è normale.

ES:

$$A = \begin{bmatrix} 0 & M & & \\ & 0 & M & \\ & & 0 & M \\ & & & 0 \end{bmatrix} \qquad A^2 = \begin{bmatrix} 0 & 0 & M^2 & 0 \\ & 0 & 0 & M^2 \\ & & 0 & 0 \\ & & & 0 \end{bmatrix} \qquad A^3 = \begin{bmatrix} 0 & & & M^3 \\ & 0 & & \\ & & 0 & \\ & & & 0 \end{bmatrix} \qquad A^4 = 0$$

Nilpotente, ma con grossa crescita intermedia.

Se A normale, $\|A^k\| = \rho(A)^k$, un poco meglio, però non risolve i problemi.

$$A^k = (V\Lambda V^{-1})^k = V\,\Lambda^k\,V^{-1}$$

$$\|A^k\| = \|\Lambda^k\| \quad \text{(se } V \text{ ortogonale)}$$

$$= \rho(A)^k$$

Idea: usiamo la forma di Schur di A.

$$A = UTU^*, \quad f(A) = U f(T) U^*$$

Ci basta saper calcolare $f(T)$, T triangolare.

Es: $n=2$

$$T = \begin{bmatrix} t_{11} & t_{12} \\ 0 & t_{22} \end{bmatrix} \qquad f(T) = \begin{bmatrix} s_{11} & s_{12} \\ \boxed{0} & s_{22} \end{bmatrix}$$

↳ perché ρ(Triangolare)

$$s_{11} = \rho(t_{11}) = f(t_{11})$$

$$s_{22} = f(t_{22})$$

Per calcolare s_{12}, uso il fatto che T e $f(T)$ commutano:

$$\begin{bmatrix} t_{11} & t_{12} \\ 0 & t_{22} \end{bmatrix}\begin{bmatrix} s_{11} & s_{12} \\ 0 & s_{22} \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} \\ 0 & s_{22} \end{bmatrix}\begin{bmatrix} t_{11} & t_{12} \\ 0 & t_{22} \end{bmatrix}$$

$$t_{11}s_{12} + t_{12}s_{22} = s_{11}t_{12} + s_{12}t_{22}.$$

$$s_{12} = \frac{s_{11}t_{12} - t_{12}s_{22}}{t_{11} - t_{22}} = t_{12} \cdot \boxed{\frac{s_{11} - s_{12}}{t_{11} - t_{22}}}$$

↳ rapporto incrementale $\dfrac{f(t_{11}) - f(t_{22})}{t_{11} - t_{22}}$

Se $t_{11} - t_{22} \neq 0$, posso calcolare s_{12}.

La stessa idea funziona per matrici più grandi:

T triangolare, $S = f(T)$ $\quad TS = ST \iff$

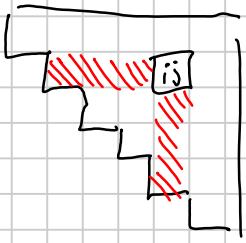
$$\sum_{i \le k \le j} t_{ik} s_{kj} = \sum_{i \le k \le j} s_{ik} t_{kj} \qquad \forall i,j$$

Da questa equazione, posso ricavare s_{ij}:

$$t_{ii} s_{ij} - s_{ij} t_{jj} = \underbrace{\sum_{i \le k < j} s_{ik} t_{kj} - \sum_{i < k \le j} t_{ik} s_{kj}}_{R}$$

$$s_{ij} = \frac{R}{t_{ii} - t_{jj}}$$

R contiene entrate di S che stanno a sinistra oppure sotto a s_{ij}.



Nell'ordine opportuno (per esempio, per sopradiagonali) riesco a calcolare tutti gli elementi di S, per sostituzione.

Ricorda Bartels-Stewart, e in effetti stiamo risolvendo

$$TS - ST = 0 \qquad \text{con T nota, S ignota}$$

(non la soluzione unica, ma per gli elementi diagonali usiamo un'altra strategia).

Ricorsione di Parlett.

Scriviamo in Matlab, calcolando in quest'ordine:

1	3	6	10	15
	2	5	9	14
		4	8	13
			7	12
				11

Problema: instabilità se $t_{ii} \approx t_{jj}$

Anche se evitiamo problemi dovuti a $\mathcal{K}(V)$, autovalori vicini causano problemi per questo algoritmo.

Soluzione: applichiamo l'algoritmo a blocchi, cercando di mettere nello stesso blocco autovalori vicini:

$$T = \begin{bmatrix} 1.99 & * & * & - & - & * \\ & 2 & & & & \vdots \\ & & 3.99 & & & \vdots \\ & & & 4.01 & & \vdots \\ & & & & 4.02 & * \\ & & & & & 6 \end{bmatrix}$$

Step 1: calcolo il valore di $f(T_{jj})$ per ogni blocco diagonale T_{jj}

Step 2: calcolo T_{ij} (per $i<j$) con una versione a blocchi della ricorsione di Parlett.

Per step 1, è naturale calcolare $f(T_{jj})$ con una serie di Taylor centrata in $\alpha = \frac{1}{k_j} \mathrm{Tr}(T_{jj})$.

Per step 2, notiamo che la versione a blocchi delle equazioni che abbiamo vista sopra è

$$T_{ii}S_{ij} - S_{ij}T_{jj} = \underbrace{\sum_{i<k<j} S_{ik}T_{kj} - \sum_{i<k<j} T_{ik}S_{kj}}_{\text{noto dai passi precedenti}}$$

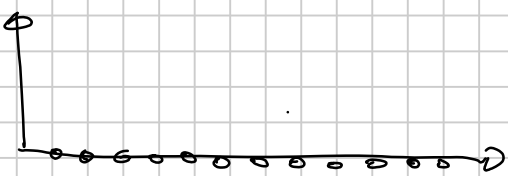
Equazione di Sylvester in S_{ij}, risolubile se $\Lambda(T_{ii}) \cap \Lambda(T_{jj})$ sono disgiunti.

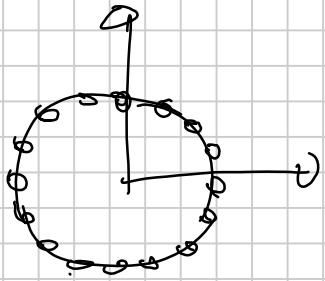
Per avere stabilità in step 2, vogliamo che $\Lambda(T_{ii})$, $\Lambda(T_{jj})$ siano il più lontani possibile.

Vogliamo blocchi ben separati con autovalori vicini.

Per alcune matrici è naturale dividere in blocchi in questo modo, ma in generale può essere problematico

Ad es.

$\Lambda(A) =$ 



Questo metodo è implementato in `fonum(A, f)` in Matlab.

Cerca di mettere nello stesso blocco autovalori che distano meno di 0.1 (soglia configurabile).

Problemi

- $|\lambda_i - \lambda_j|$ non è la misura giusta del mal condizionamento delle equazioni di Sylvester, dovrei usare $sep(T_{ii}, T_{jj})$, ma non c'è un buon modo di calcolarla in dimensione alta
- Facile trovare casi in cui gli autovalori finiscono tutti in un grosso blocco anche quando sono lontani, e Taylor converge piano.

  Oss: per calcolare d termini di una serie di Taylor su una matrice $n \times n$ mi servono $O(n^3 d)$ operazioni aritmetiche, se $d \sim n$, è più che cubico come costo.

- Per usare sviluppi di Taylor, mi servono le derivate di f!

funzione() vuole in input $f(x,k)$ che calcola $f^{(k)}(x)$.
(a parte poche funzioni "hardcoded")