UNIVERSITÀ DI PISA

Corso di Laurea Triennale in Matematica

Tesi di Laurea Triennale

Metodi Numerici per Radici p-esime di Matrici

Candidato:

Greta Malaspina

Relatore:

Prof.ssa Beatrice Meini

Anno Accademico 2014/2015

Indice

Introduzione

Lo scopo di questa tesi è presentare alcuni metodi numerici per il calcolo di radici p-esime di matrici. Data $A \in \mathbb{C}^{n\times n}$ senza autovalori reali negativi e un intero p, si definisce radice p-esima di A ogni matrice Y che risolve l'equazione $X^p = A$. La radice p-esima di matrice può essere interpretata come una particolare funzione di matrici.
Nel primo capitolo introdurremo le nozioni fondamentali riguardo le funzioni di matrici. In letteratura, data una funzione f e una generica matrice A, esistono diverse definizioni di $f(A)$ che passano attraverso la forma di Jordan della matrice, l'interpolazione polinomiale e la formula integrale di Cauchy: presenteremo tali definizioni e daremo una dimostrazione della loro equivalenza, dopodiché enunceremo e dimostreremo alcuni risultati che ci serviranno nel resto del lavoro. Nel secondo capitolo presenteremo alcuni metodi per il calcolo di radici p-esime. Tali metodi si dividono principalmente in due categorie: metodi ricorsivi e metodi di approssimazione. Tutti gli algoritmi che vedremo, inoltre, prevedono come passaggio preliminare il calcolo della forma normale di Schur della matrice di partenza, che conduce a una matrice triangolare o triangolare a blocchi con blocchi diagonali di dimensione massima 2.
Presenteremo tre algoritmi ricorsivi che calcolano la radice Y della forma normale di Schur, procedendo in modo ricorsivo blocco per blocco, seguendo lo stesso schema generale: calcolano direttamente la radice p-esima dei blocchi diagonali, dopodiché seguono un procedimento ricorsivo per calcolare la sottomatrice restante, utilizzando ad ogni passo i blocchi calcolati nei passi precedenti e risolvendo un'equazione matriciale riconducibile ad un sistema lineare. Ciò che distingue i tre metodi è proprio la forma dell'equazione matriciale. Il metodo di Smith [12] usa tutte le potenze di Y fino alla $(p-1)$-esima, e l'algoritmo che ne deriva richiede tempo asintotico $O(n^3p + n^2p^2)$. Il metodo di Greco-Iannazzo [5] sfrutta un procedimento di esponenziazione binaria che permette di ridurre il costo totale del metodo a $O(n^3 \log_2 p + n^2 p)$ operazioni. Il metodo di Iannazzo-Manasse [10] apporta alcune modifiche all'equazione che collega i blocchi di Y a quelli della matrice di partenza e utilizza un algoritmo efficiente per il calcolo dei coefficienti di tale equazione. Il calcolo della radice p-esima secondo questo procedimento ha costo $O(n^3 \log_2 p)$.
Nella seconda parte del secondo capitolo presenteremo alcuni metodi basati sull'algoritmo di Schur-Padé [6]. Inizieremo dimostrando alcuni risultati fondamentali riguardo la teoria delle approssimanti di Padé e introdurremo le funzioni ipergeometriche ${}_2F_1$. Tali

funzioni saranno importanti nella derivazione del metodo perché, sfruttando l'uguaglianza $(1-x)^{1/p} = {}_2F_1(-1/p, 1; 1; x)$, troveremo una rappresentazione dell'approssimante di Padé $\mathcal{R}_m$ di $(1-x)^{1/p}$ come troncamento di una frazione continua che può essere valutata in modo efficiente. L'analisi dell'errore di approssimazione induce lo schema generale dell'algoritmo: data T matrice triangolare superiore a blocchi e $p \in (-1, 1)$ estraiamo radici quadrate di T fino ad avere $T^{1/2^k} \simeq I$ dopodiché valutiamo $\mathcal{R}_m$ in $I - T^{1/2^k}$ e eleviamo la matrice ottenuta alla potenza 2^k. Concluderemo il capitolo presentando alcuni miglioramenti del metodo di Schur-Padé [7], tra cui l'utilizzo di sola aritmetica reale nel caso in cui la matrice di partenza sia reale e un'estensione che permette di calcolare, simultaneamente a T^p anche le sue derivate di Fréchet.
Nel terzo capitolo riporteremo i risultati di alcuni test effettuati sull'algoritmo di Iannazzo-Manasse e Schur-Padé. In particolare confronteremo i tempi di esecuzione dei due metodi al variare di p e della dimensione della matrice, l'accuratezza del risultato ottenuto per matrici mal condizionate e, nel caso del metodo di Iannazzo-Manasse, l'accuratezza nel calcolo delle determinazioni non principali della radice.

Capitolo 1

Funzioni di Matrici

Lo scopo di questo capitolo sarà chiarire cosa intendiamo per *funzioni di matrici*, proporne diverse definizioni e dimostrarne l'equivalenza presentando alcune proprietà che ci saranno utili nel resto del lavoro. Nell'ultimo paragrafo considereremo brevemente il caso particolare in cui le funzioni considerate siano radici p-esime. Per quanto riguarda il contenuto di questo capitolo facciamo riferimento ad Higham [9].

1.1 Definizioni

Data una funzione scalare f definita dall'insieme $\mathbb{C}$ dei numeri complessi in sè, vogliamo estendere la sua definizione allo spazio delle matrici quadrate, in modo tale che l'immagine $f(A)$ di una matrice A in $\mathbb{C}^{n\times n}$ sia ancora una matrice nello stesso spazio.

1.1.1 Definizione mediante forma canonica di Jordan

Data una matrice A in $\mathbb{C}^{n\times n}$, sappiamo che può essere espressa in forma canonica di Jordan, ovvero che esiste una matrice Z invertibile tale che $Z^{-1}AZ = J = \mathrm{diag}(J_1, \ldots, J_p)$ dove

$$J_k = \begin{pmatrix} \lambda_k & 1 & & \\ & \lambda_k & \ddots & \\ & & \ddots & 1 \\ & & & \lambda_k \end{pmatrix} \in \mathbb{C}^{m_k \times m_k}$$

con $\lambda_k \in \mathrm{Spec}(A),\ \ k = 1, \ldots, p$.
Consideriamo un generico autovalore λ_k della matrice e supponiamo che la funzione f ammetta sviluppo di Taylor di centro λ_k

$$f(t) = \sum_{j\geq 0} \frac{1}{j!} f^{(j)}(\lambda_k)(t - \lambda_k)^j.$$

Sia J_k un blocco di Jordan di ordine m_k relativo a λ_k, abbiamo $J_k = \lambda_k I + N_k$, dove N_k è la matrice con elementi uguali a 1 sulla prima sopradiagonale e zero altrove e I è la matrice identità di ordine m_k. Valutando lo sviluppo di Taylor in J_k, otteniamo:

$$f(J_k) = \sum_{j\geq 0} \frac{1}{j!} f^{(j)}(\lambda_k I)(J_k - \lambda_k I)^j = \sum_{j\geq 0} \frac{1}{j!} f^{(j)}(\lambda_k I)(N_k)^j =$$

poiché la matrice N_k è nilpotente di indice m_k

$$= \sum_{j=0}^{m_k-1} \frac{1}{j!} f^{(j)}(\lambda_k I)(N_k)^j = \begin{pmatrix} f(\lambda_k) & f'(\lambda_k) & \dots & \frac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{pmatrix}.$$

Definizione 1.1. Sia $\mathrm{Spec}(A) = \{\lambda_1, \dots, \lambda_s\}$ lo spettro della matrice A e sia n_k l'indice dell'autovalore λ_k, ovvero il massimo delle dimensioni dei blocchi di Jordan relativi a λ_k. Una funzione f si dice *definita sullo spettro di* A se esistono i valori

$$f^{(j)}(\lambda_k) \quad \text{per} \quad k = 1, \dots, s \quad \text{e} \quad j = 0, \dots, (n_k - 1).$$

A questo punto possiamo dare la prima definizione di $f(A)$.

Definizione 1.2. Sia f funzione scalare definita sullo spettro di $A \in \mathbb{C}^{n\times n}$ con forma di Jordan $Z^{-1}AZ = J = \mathrm{diag}(J_1, \dots, J_p)$. Definiamo:

$$f(A) := Z\,\mathrm{diag}(f(J_1), \dots, f(J_p))Z^{-1}$$

dove, per ogni blocco di Jordan J_k, poniamo

$$f(J_k) := \begin{pmatrix} f(\lambda_k) & f'(\lambda_k) & \dots & \frac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & f(\lambda_k) & \ddots & \vdots \\ & & \ddots & f'(\lambda_k) \\ & & & f(\lambda_k) \end{pmatrix}.$$

Affinché la definizione abbia senso nel caso in cui la funzione f ammetta più di una determinazione, è necessario considerare la stessa determinazione per ognuno dei blocchi di Jordan, ma questa può differire da un blocco all'altro. Nel seguito richiederemo però una condizione aggiuntiva, ovvero che nel caso in cui siano presenti diversi blocchi di Jordan relativi allo stesso autovalore, per ognuno di questi blocchi venga scelta la stessa determinazione di f. Una funzione di matrici che verifichi quest'ultima proprietà sarà detta *primaria*.

Osservazione. Se la matrice A è diagonalizzabile, ovvero $A = ZDZ^{-1}$ con D matrice diagonale i cui elementi diagonali sono proprio gli autovalori di A, dalla definizione appena data abbiamo $f(A) = Z\,\mathrm{diag}(f(\lambda_1), \ldots, f(\lambda_n))Z^{-1}$. In questo caso, quindi, $f(A)$ è una matrice i cui autovalori sono le immagini degli autovalori della matrice di partenza e per ogni λ nello spettro di A gli autovettori relativi a $f(\lambda)$ per $f(A)$ sono proprio gli autovettori relativi a λ per A.

1.1.2 Definizione mediante interpolazione polinomiale

Data $A \in \mathbb{C}^{n\times n}$ il suo *polinomio minimo* $q_A(t)$ è definito come il polinomio monico di grado minimo che si annulla in A e sappiamo che, con le stesse notazioni del paragrafo precedente, q_A verifica le seguenti proprietà:

- se p è un polinomio che si annulla in A, allora q_A divide p
- $q_A(t) = \prod_{k=1}^{s}(t-\lambda_k)^{n_k}$, in particolare q_A si annulla su tutti gli autovalori di A.

Vogliamo ora dimostrare che dato un polinomio p, la matrice $p(A)$ è completamente determinata dai valori di p sullo spettro di A.

Teorema 1.1.
Dati due polinomi p_1 e p_2, $p_1(A) = p_2(A) \iff p_1(\lambda) = p_2(\lambda)\ \forall \lambda \in \mathrm{Spec}(A)$.

Dimostrazione. Sia $d(t) = p_1(t) - p_2(t)$ il polinomio differenza.
Se $p_1(A) = p_2(A)$ allora $d(A) = 0$, e per le proprietà appena ricordate abbiamo che q_A divide d e $d(\lambda) = 0\ \forall \lambda$ autovalore di A, ovvero $p_1(\lambda) = p_2(\lambda)$ su tutto la spettro di A.
Viceversa, se $p_1(\lambda) = p_2(\lambda)\ \forall \lambda \in \mathrm{Spec}(A)$ allora ogni autovalore di A è radice di $d(t)$ e quindi d è multiplo di q_A, e questo implica che $d(A) = 0$, ovvero che $p_1(A) = p_2(A)$.
□

Definizione 1.3. Sia f funzione scalare definita sullo spettro di $A \in \mathbb{C}^{n\times n}$ e sia q_A il suo polinomio minimo. Chiamiamo *polinomio di interpolazione di Hermite* per f e A il polinomio $p \in \mathbb{C}[t]$ che verifica le seguenti proprietà:

- $deg(p) \leq deg(q_A) = \sum_{k=1}^{s} n_k$
- $p^{(j)}(\lambda_k) = f^{(j)}(\lambda_k) \quad \forall \lambda_k \in \mathrm{Spec}(A)$ e $\forall j = 0 : n_k - 1$

Vediamo come ottenere tale polinomio esplicitamente, date la funzione f e la matrice A. Se la matrice ha tutti autovalori distinti, e quindi $s = n$ e $n_k = 1$ per ogni k, possiamo usare la formula di interpolazione di Lagrange, ottenendo:

$$p(t) = \sum_{i=1}^{n} f(\lambda_i) \prod_{j=1 j\neq i}^{n} \frac{t-\lambda_j}{\lambda_i - \lambda_j}.$$

Nel caso generale invece, si verifica che p è dato da:

$$p(t) = \sum_{i=1}^{s} \left[\left(\sum_{j=0}^{n_i-1} \frac{\Phi_i^{(j)}(\lambda_i)}{j!}(t-\lambda_i)^j \right) \prod_{j \neq i} (t-\lambda_j)^{n_j} \right]$$

dove

$$\Phi_i(t) = \frac{f(t)}{\prod_{j \neq i}(t-\lambda_j)^{n_j}}.$$

Definizione 1.4. Data $A \in \mathbb{C}^{n \times n}$ e f funzione definita sullo spettro di A, definiamo $f(A) := p(A)$, dove p è il polinomio di interpolazione di Hermite per la coppia $f,\ A$.

Osservazione. Il polinomio di interpolazione dipende strettamente dallo spettro della matrice A di cui vogliamo calcolare l'immagine, la definizione che abbiamo appena dato non significa, quindi, che $f(M) = p(M)$ per ogni matrice in $\mathbb{C}^{n \times n}$. Si dimostra facilmente però che, date due matrici A e B tali che $\mathrm{Spec}(B) \subseteq \mathrm{Spec}(A)$, se $f(A) = p(A)$ allora $f(B) = p(B)$.

1.1.3 Definizione mediante formula integrale di Cauchy

Sia f una funzione a valori complessi definita su un aperto di $\mathbb{C}$ e sia Γ una curva semplice chiusa contenuta nell'aperto. Sappiamo che, per ogni punto a appartenente alla regione interna a Γ vale l'uguaglianza

$$f(a) = \frac{1}{2\pi \mathrm{i}} \int_\Gamma \frac{f(z)}{z-a} dz$$

e, più in generale,

$$f^{(n)}(a) = \frac{n!}{2\pi \mathrm{i}} \int_\Gamma \frac{f(z)}{(z-a)^{n+1}} dz.$$

Usiamo queste relazioni per dare una nuova definizione di $f(A)$, ricordando che l'integrale di una matrice si intende componente per componente.

Definizione 1.5. Data $A \in \mathbb{C}^{n \times n}$ e f funzione analitica in un intorno dello spettro di A, definiamo:

$$f(A) := \int_\Gamma f(z)(zI - A)^{-1} dz$$

dove Γ è una curva semplice chiusa sull'aperto su cui f è analitica, che contiene lo spettro di A.

1.2 Equivalenze e proprietà

Teorema 1.2.
Per ogni funzione scalare f le definizioni 1.2 e 1.4 sono equivalenti. Se f è analitica, anche la definizione 1.5 è equivalente alle precedenti.

Dimostrazione. Sia $A \in \mathbb{C}^{n\times n}$, $\mathrm{Spec}(A) = \{\lambda_1, \dots, \lambda_p\}$ e $A = ZJZ^{-1}$ la sua forma di Jordan. Mostriamo l'equivalenza tra 1.2 e 1.4.
Per la definizione 1.2

$$f(A) = Z\,\mathrm{diag}(f(J_1), \dots, f(J_s))Z^{-1}$$

mentre per 1.4, detto p il polinomio di interpolazione

$$f(A) = p(A) = Zp(J)Z^{-1} = Z\,\mathrm{diag}(p(J_1), \dots, p(J_s))Z^{-1}$$

quindi ci basta dimostrare che per la definizione 1.2 $f(J_k) = p(J_k)$ su ognuno dei blocchi di Jordan.
Abbiamo visto che $f(J_k) = q(J_k)$ dove q è il polinomio dato da

$$q(t) = \sum_{j\geq 0}^{m_k-1} \frac{1}{j!} f^{(j)}(\lambda_k)(t-\lambda_k)^j,$$

quindi $q(\lambda_k) = f(\lambda_k) = p(\lambda_k)$, ovvero p e q coincidono su l'unico autovalore di J_k. Per il Teorema 1.1 abbiamo allora $p(J_k) = q(J_k)$ e questo ci da diretteamente la tesi.
Mostriamo ora l'equivalenza tra 1.5 e 1.2.
Consideriamo l'operatore $\Phi(A) := \frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z)(zI - A)^{-1}dz$, che è $f(A)$ per la definizione 1.5 ed è continuo. Se A è una matrice diagonalizzabile con $A = MDM^{-1}$, allora

$$\Phi(A) = \frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z)(zI - MDM^{-1})^{-1}dz = M\frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z)(zI - D)^{-1}dzM^{-1} =$$

poiché su ogni componente non diagonale la matrice integranda è nulla

$$= M\,\mathrm{diag}\left(\frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z)(z-\lambda_1)^{-1}dz, \dots\dots, \frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z)(z-\lambda_p)^{-1}dz\right) M^{-1} =$$

per la formula integrale di Cauchy

$$= M\,\mathrm{diag}(f(\lambda_1), \dots, f(\lambda_p))M^{-1}$$

Che è proprio la definizione 1.2 di $f(A)$. Nel caso in cui A abbia forma di Jordan MJM^{-1} con J non diagonale, si ottiene la tesi seguendo un procedimento analogo al precedente e utilizzando la formula integrale di Cauchy anche per le derivate della funzione f.

□

Dimostriamo adesso alcune proprietà delle funzioni di matrici che ci serviranno nei prossimi capitoli.

Lemma 1.1. Data $A \in \mathbb{C}^{n\times n}$, se f è una funzione definita sullo spettro di A, allora:

1. le funzioni di matrici sono *invarianti per similitudine*, ovvero:
   per ogni matrice M invertibile, $f(MAM^{-1}) = Mf(A)M^{-1}$

2. se $\mathrm{Spec}(A) = \{\lambda_1, \ldots, \lambda_p\}$ allora $\mathrm{Spec}(f(A)) = \{f(\lambda_1), \ldots, f(\lambda_p)\}$

3. se A è triangolare a blocchi con blocchi diagonali $A_{11}, \ldots, A_{mm}$ allora $f(A)$ ha la stessa struttura di A e i suoi blocchi diagonali sono $f(A_{11}), \ldots, f(A_{mm})$

4. $f(I_m \otimes A) = I_m \otimes f(A)$

5. $f(A \otimes I_m) = f(A) \otimes I_m$

Dimostrazione. 1. Segue direttamente dal fatto che per la definizione 1.4 $f(A)$ è un polinomio in A.

2. Per la definizione 1.2 se A ha forma di Jordan $Z\,\mathrm{diag}(J_1, \ldots, J_s)Z^{-1}$ allora $f(A) = Z\,\mathrm{diag}(f(J_1), \ldots, f(J_s))Z^{-1}$ con

$$f(J_k) := \begin{pmatrix} f(\lambda_k) & \ldots & \frac{f^{(m_k-1)}(\lambda_k)}{(m_k-1)!} \\ & \ddots & \vdots \\ & & f(\lambda_k) \end{pmatrix}$$

quindi ogni blocco diagonale $f(J_k)$ ha come unico autovalore proprio $f(\lambda_k)$, e questi saranno anche gli autovalori di $f(A)$.

3. Segue direttamente dal fatto che $f(A)$ è un polinomio in A e che per ogni m naturale A^m è ancora una matrice triangolare a blocchi i cui blocchi diagonali sono proprio le potenze $m-$esime degli A_{kk}.

4. $I_m \otimes A$ è una matrice diagonale a blocchi con m blocchi diagonali tutti uguali ad A, quindi $f(I_m \otimes A) = f(\mathrm{diag}(A, \ldots, A))$ che, per il punto precedente, è proprio $\mathrm{diag}(f(A), \ldots, f(A)) = I_m \otimes f(A)$.

5. Sappiamo che esiste una matrice di permutazione P tale che $A \otimes I_m = P(I_m \otimes A)P^{-1}$. Per quanto visto fino ad ora abbiamo, quindi: $f(A \otimes I_m) = f(P(I_m \otimes A)P^{-1})) = Pf((I_m \otimes A))P^{-1} = P(I_m \otimes f(A))P^{-1} = f(A) \otimes I_m$.

□

Vediamo adesso un risultato riguardo il raggio di convergenza della serie di Taylor di una funzione di matrici f, per la dimostrazione rimandiamo a [9].

Teorema 1.3. Sia f una funzione scalare con sviluppo di Taylor di centro α dato da $f(z) = \sum_{k=0}^{+\infty} a_k(z-\alpha)^k$ dove $a_k = \frac{1}{k!}f^{(k)}(\alpha)$, con raggio di convergenza r. Data una matrice $A \in \mathbb{C}^{n\times n}$, sono fatti equivalenti:

1. esiste $f(A)$ e vale $f(A) = \sum_{k=0}^{+\infty} a_k(A-\alpha I)^k$;

2. $\forall \lambda_k \in \mathrm{Spec}(A)$ vale una tra le seguenti condizioni:

- $|\lambda_k - \alpha| < r$;
- $|\lambda_k - \alpha| = r$ e la serie di Taylor di $f^{(n_k-1)}(z)$ converge nel punto $z = \lambda_k$.

1.3 Radici primarie di matrici

Nel seguito del nostro lavoro vogliamo studiare il caso particolare in cui la funzione f sia una radice p-esima, ovvero una funzione $f : \Omega \subseteq \mathbb{C} \longrightarrow \mathbb{C}$ tale che $f(z)^p = z$. Se Ω ha s componenti connesse, nessuna delle quali contiene 0 o un cammino chiuso attorno a 0, allora esistono esattamente p^s funzioni *radici p-esime* analitiche in Ω, ottenute prendendo su ogni componente connessa una delle possibili determinazioni della radice.
Data una matrice $A \in \mathbb{C}^{n\times n}$ non singolare consideriamo $\Omega := \mathrm{Spec}(A) = \{\lambda_1, \ldots, \lambda_s\}$, come abbiamo appena osservato esistono p^s radici p-esime ottenute scegliendo una determinazione della radice per ogni autovalore. Ognuna di queste radici è una funzione primaria definita sullo spettro di A e si estende quindi ad una funzione di matrici, secondo le definizioni che abbiamo dato nei paragrafi precedenti. Scelta una funzione $f(z)$ radice p-esima, abbiamo quindi $f(A)$, che sarà una soluzione dell'equazione matriciale $X^p = A$.
Data $A \in \mathbb{R}^{n\times n}$ non singolare con autovalori $\mathrm{Spec}(A) = \{\lambda_1, \ldots, \lambda_s\}$, se la radice primaria f scelta è tale che $f(\bar{z}) = \overline{f(z)}$ $\forall z \in \mathrm{Spec}(A)$, ovvero in modo che le radici degli autovalori reali siano reali e che la determinazione scelta per ogni coppia di autovalori complessi coniugati sia la stessa, allora $f(A)$ sarà ancora una matrice reale.

Capitolo 2

Metodi Numerici

Lo scopo di questo capitolo è presentare alcuni metodi numerici per il calcolo di radici primarie di matrici. Gli algoritmi esistenti si dividono principalmente in due categorie:

- *algoritmi ricorsivi di Schur* che dopo aver trasformato la matrice di partenza in forma di Schur, ne calcolano le radici attraverso una formula diretta;
- *algoritmi di approssimazione o di iterazione razionale* che calcolano approssimazioni razionali delle radici della matrice, o iterazioni razionali convergenti a tali radici.

Vedremo ora alcuni metodi appartenenti alle due classi, soffermandoci sulla loro derivazione. In particolare descriveremo l'algoritmo di Iannazzo-Manasse per quel che riguarda i metodi di Schur, e l'algoritmo di Schur-Padé per quel che riguarda i metodi di approssimazione.

2.1 Algoritmo di Iannazzo-Manasse

Sia $A \in \mathbb{C}^{n\times n}$ invertibile con autovalori $\{\lambda_1, \ldots, \lambda_s\}$ non avente autovalori reali negativi e $f(z)$ una radice p-esima primaria definita sullo spettro di A. Per il Lemma 1.1 sappiamo che, se $A = QTQ^H$ con Q matrice unitaria e T triangolare superiore a blocchi, allora $f(A) = Qf(T)Q^H$ e possiamo quindi ridurci a calcolare $f(T)$ sfruttando il fatto che T e $f(T)$ hanno la stessa struttura. In particolare considereremo due casi:

- T è la matrice triangolare superiore ottenuta calcolando la forma normale di Schur di A. Sappiamo che gli elementi sulla diagonale di T sono proprio gli autovalori della matrice di partenza.
- A è reale e T è la matrice quasi triangolare superiore a blocchi ottenuta calcolando la forma di Schur reale di A. T avrà blocchi diagonali di ordine 1 oppure 2, che corrisponderanno rispettivamente agli autovalori reali o alle coppie di autovalori complessi coniugati di A. In questo caso, chiediamo che anche la radice calcolata sia reale.

Inizieremo presentando brevemente due metodi per il caso reale, l'algoritmo di Smith [12] e di Greco-Iannazzo [5], e descriveremo più nel dettaglio l'algoritmo di Iannazzo-Manasse.

2.1.1 Metodo di Smith

Data $T \in \mathbb{R}^{n\times n}$ quasi triangolare superiore con blocchi T_{ij} per $i, j = 1, \dots, s$ e $f(z)$ radice p-esima primaria tale che $f(T)$ sia ancora una matrice reale, vogliamo calcolare $Y := f(T)$ che sarà quindi una matrice che verifica l'equazione $Y^p = T$.
Costruiamo un insieme di matrici con la stessa struttura a blocchi di T, $\{R^{(0)}, \dots, R^{(p-1)}\}$ date da $R^{(k)} := Y^{k+1}$che verificheranno quindi $R^{(0)} = Y$, $R^{(p-1)} = T$ e $R^{(k)} = YR^{(k-1)}$. Vediamo come usare queste uguaglianze per mettere in relazione i blocchi della matrice T con quelli di Y e delle $R^{(k)}$.
Per ogni $i < j$ abbiamo

$$T_{ij} = (R^{(p-1)})_{ij} = (YR^{(p-2)})_{ij} = Y_{ii}R^{(p-2)}_{ij} + Y_{ij}R^{(p-2)}_{jj} + \sum_{k=i+1}^{j-1} Y_{ik}R^{(p-2)}_{kj}$$

e i blocchi di ognuna delle $R^{(l)}$ si possono scrivere ricorsivamente come

$$\begin{aligned} R^{(l)}_{ij} &= (YR^{(l-1)})_{ij} = Y_{ii}R^{(l-1)}_{ij} + Y_{ij}R^{(l-1)}_{jj} + \sum_{k=i+1}^{j-1} Y_{ik}R^{(l-1)}_{kj} = \\ &= \sum_{h=0}^{l} Y_{ii}^{l-h}Y_{ij}Y_{jj}^{h} + \sum_{h=0}^{l-1} Y_{ii}^{l-h-1}B^{(l)}_{ij} \end{aligned} \tag{2.1}$$

dove l'ultima uguaglianza si può dimostrare per induzione e le matrici B sono date da $B^{(l)}_{ij} = \sum_{k=i+1}^{j-1} Y_{ik}R^{(l)}_{kj}$. Sostituendo quest'ultima uguaglianza nella prima, otteniamo:

$$T_{ij} = \sum_{l=0}^{p-1} Y_{ii}^{p-l-1}Y_{ij}Y_{jj}^{l} + \sum_{l=0}^{p-2} Y_{ii}^{p-l-2}B^{(l)}_{ij} \tag{2.2}$$

che è l'equazione che sta alla base del metodo di Smith.
Nell'uguaglianza compaiono: il blocco T_{ij} della matrice di partenza, i blocchi diagonali Y_{kk} e i blocchi Y_{ik}, $R^{(l)}_{kj}$ per $i \le k < j$ ovvero i blocchi che stanno in basso a sinistra rispetto alla posizione (i, j) del blocco che stiamo considerando. Supponendo di conoscere quest'ultimo gruppo di blocchi, poiché i blocchi diagonali sono facilmente calcolabili, l'equazione (2.2) non è altro che un'equazione matriciale da cui possiamo ricavare Y_{ij}. Quello che faremo sarà quindi usare l'equazione (2.2) per calcolare i blocchi di Y colonna per colonna, dal basso verso l'alto e da sinistra verso destra.
Per ogni j indice di colonna fissato:

1. calcoliamo Y_{jj} radice di T_{jj}

- Se T_{jj} è un blocco 1×1, ovvero $T_{jj} = (\lambda_j)$ con λ_k autovalore della nostra matrice di partenza, chiaramente avremo $Y_{jj} = (f(\lambda_j))$;
- se invece $T_{jj} = \begin{pmatrix} \theta & \mu \\ -\mu & \theta \end{pmatrix}$ con $\theta \pm i\mu$ coppia di autovalori complessi coniugati, avremo $T_{jj} = \begin{pmatrix} \alpha & \beta \\ -\beta & \alpha \end{pmatrix}$ con $\alpha + i\beta = f(\theta + i\mu)$

2. per $k = 0 : p-2$ calcoliamo $R_{jj}^{(k)} = Y_{jj} R_{jj}^{(k-1)}$
3. per $i = j-1 : 1$ ricaviamo Y_{ij} dall'equazione (2.2) e calcoliamo $R_{ij}^{(k)}$ per $k = 1 : p-2$ usando (2.1).
   Usando l'operatore lineare vec che, concatenando le colonne, porta una matrice in un vettore, e il fatto che $\operatorname{vec}(AMB) = (B^T \otimes A)\operatorname{vec}(M)$, possiamo trasformare l'equazione (2.2) in un sistema lineare:

$$\operatorname{vec}\left(T_{ij} - \sum_{l=0}^{p-2} Y_{ii}^{p-l-2} B_{ij}^{(l)}\right) = \operatorname{vec}\left(\sum_{l=0}^{p-1} Y_{ii}^{p-l-1} Y_{ij} Y_{jj}^{l}\right) = \sum_{l=0}^{p-1} \left((Y_{jj}^{l})^T \otimes Y_{ii}^{p-l-1}\right) \operatorname{vec}(Y_{ij}).$$

   Otteniamo quindi

$$\left(\sum_{l=0}^{p-1} (Y_{jj}^{l})^T \otimes Y_{ii}^{p-l-1}\right) \operatorname{vec}(Y_{ij}) = \operatorname{vec}\left(T_{ij} - \sum_{l=0}^{p-2} Y_{ii}^{p-l-2} B_{ij}^{(l)}\right),$$

   si può dimostrare che questo ammette soluzione unica e possiamo quindi ricavare Y_{ij}.

Il costo computazionale dell'algoritmo appena descritto è $O(n^3 p + n^2 p^2)$, un'analisi completa della complessità si può trovare nell'articolo originale [12].

2.1.2 Metodo di Greco-Iannazzo

Il metodo di Greco-Iannazzo deriva direttamente da quello di Smith e si basa sull'osservazione che usando un procedimento di *binary powering* non è necessario calcolare tutte le p potenze di Y calcolate nell'algoritmo di Smith.

Consideriamo l'espansione binaria di $p = \sum_{i=0}^{\lfloor \log_2 p \rfloor} b_i 2^i$ con $b_i \in \{0, 1\}$ e $b_{\lfloor \log_2 p \rfloor} = 1$, chiamiamo $c(p)$ l'insieme delle posizioni delle cifre non zero dello sviluppo e $m+1$ la sua cardinalità, ovvero $c(p) := \{0 \le k \le \lfloor \log_2 p \rfloor | b_k = 1\} = \{c_0, \ldots, c_m\}$ con $c_0 = \lfloor \log_2 p \rfloor$ e $c_k = \max\{h < c_{k-1} | b_h = 1\}$. Ora definiamo due insiemi di matrici $\{V^{(k)}\}_k$ per $k = 0 : c_0$ e $\{W^{(k)}\}_k$ per $k = 0 : m$ dati da

$$\begin{cases} V^{(0)} = Y \\ V^{(k)} = V^{(k-1)} V^{(k-1)} \end{cases} \qquad \begin{cases} W^{(0)} = V^{(c_0)} \\ W^{(k)} = W^{(k-1)} V^{(c_k)} \end{cases} \tag{2.3}$$

Da queste relazioni, poiché ovviamente $V^{(\lfloor \log_2 p \rfloor)} = Y^{2^k}$ e $W^{(m)} = Y^p = T$, possiamo ricavare un'equazione simile a quella usata nell'algoritmo di Smith, che useremo per ricalcarne i passi. Per $k \geq 1$ definiamo la seguente successione di insiemi:

$$A_1 = \{(0,1,0)\} \quad A_k = \bigcup_{(r,s,t)\in A_{k-1}} \left\{(r+2^{k-1}, s, t), (r, s, t+2^{k-1})\right\} \cup \{(0,k,0)\}.$$

Osserviamo per iniziare che le due successioni di matrici verificano:

$$\begin{aligned} V_{ij}^{(k)} &= V_{ii}^{(k-1)} V_{ij}^{(k-1)} + V_{ij}^{(k-1)} V_{jj}^{(k-1)} + B_{ij}^{(k)} \\ W_{ij}^{(h)} &= W_{ii}^{(h-1)} V_{ij}^{(c_h)} + W_{ij}^{(h-1)} V_{jj}^{(c_h)} + C_{ij}^{(h)} \end{aligned} \tag{2.4}$$

dove $B_{ij}^{(h)} = \sum_{k=i+1}^{j-1} V_{ik}^{(h-1)} V_{kj}^{(h-1)}$ e $C_{ij}^{(h)} = \sum_{k=i+1}^{j-1} W_{ik}^{(h-1)} V_{kj}^{(c_h)}$.
Se $p = 2^c$ allora si può dimostrare per induzione su c che

$$T_{ij} = \sum_{l=0}^{p-1} Y_{ii}^{l} V_{ij}^{(0)} Y_{jj}^{p-l-1} + \sum_{(r,s,t)\in A_c} Y_{ii}^{r} B_{ij}^{(s)} Y_{jj}^{t}$$

mentre nel caso generale, per $p = \sum_{j=1}^{m} 2^{c_j}$ abbiamo

$$\begin{aligned} T_{ij} = &\sum_{h=0}^{p-1} Y_{ii}^{h} V_{ij}^{(0)} Y_{jj}^{p-h-1} + \sum_{h=i}^{m} C_{ij}^{(h)} Y_{jj}^{2^{c_{h+1}}+\cdots+c_m} + \\ &+ \sum_{h\in c(p)^+} Y_{ii}^{p-2^{c_h}-\cdots-c_m} \left(\sum_{(r,s,t)\in A_{c_h}} Y_{ii}^{r} B_{ij}^{(s)} Y_{jj}^{T} \right) Y_{jj}^{2^{c_{h+1}}+\cdots+c_m} \end{aligned} \tag{2.5}$$

A questo punto l'algoritmo procede in modo del tutto analogo a quello di Smith: per ogni j indice di colonna fissato

1. calcoliamo Y_{jj} radice di T_{jj} come già visto
2. per $k = 0 : \lfloor \log_2 p \rfloor$ e $h = 0 : m$ calcoliamo $V_{jj}^{(k)} = V_{jj}^{(k-1)} V_{jj}^{(k-1)}$ e $W_{jj}^{(h)} = W_{jj}^{(k-1)} V_{jj}^{(c_k)}$
3. per $i = j-1 : 1$ ricaviamo Y_{ij} dall'equazione (2.5) e calcoliamo $V_{ij}^{(k)}$ per $k = \lfloor \log_2 p \rfloor$ e $W_{ij}^{(h)}$ per $h = 0 : m$ usando (2.4)

L'utilizzo dell'espansione binaria di p fa sì che il costo del calcolo delle matrici ausiliarie V e W sia logaritmico rispetto a p, tuttavia la risoluzione del sistema (2.5) e in particolare il calcolo dei coefficienti richiedono ancora un numero di operazioni che dipende linearmente da p, portando ad un costo asintotico di $O(n^3 \log_2 p + n^2 p)$.

2.1.3 Metodo di Iannazzo-Manasse

Il metodo di Iannazzo-Manasse parte da una costruzione analoga a quella appena vista e ne deriva una nuova equazione che può essere valutata con $O(\log_2 p)$ operazioni, portando quindi ad un algoritmo di costo inferiore rispetto a quelli considerati fino ad ora. Per quanto riguarda la derivazione che segue facciamo riferimento all'articolo originale di B. Iannazzo e C. Manasse [10].
Per comodità ridefiniamo diversamente dal paragrafo precedente alcune delle notazioni e le due successioni di matrici $V^{(k)}$ e $W^{(k)}$. Sia $t := \lfloor \log_2 p \rfloor + 1$ il numero di cifre dello sviluppo binario di p, $p = \sum_{i=1}^{t} b_i 2^{t-i} = \sum_{j=1}^{m} 2^{\alpha_j}$ dove $b_1 = 1$, $b_i \in \{0, 1\}$, $0 \leq \alpha_m < \alpha_{m-1} < \cdots < \alpha_1$ corrispondono alle posizioni delle cifre diverse da zero dello sviluppo e m è il numero di tali cifre.
Definiamo $\{V^{(k)}\}_k$ per $k = 1 : t + 1$ e $\{W^{(k)}\}_k$ per $k = 1 : m + 1$ dati da

$$\begin{cases} V^{(1)} = I \\ V^{(2)} = Y \\ V^{(k)} = V^{(k-1)} V^{(k-1)} \end{cases} \qquad \begin{cases} W^{(1)} = I \\ W^{(k)} = W^{(k-1)} V^{(\alpha_{k-1}+2)} \end{cases} \tag{2.6}$$

e osserviamo che vale: $V^{(k)} = Y^{2^{k-2}}$, $W^{(k)} = Y^{2^{\alpha_1} + \cdots + 2^{\alpha_{k-1}}}$ e quindi
$W^{(m+1)} = Y^{2^{\alpha_1} + \cdots + 2^{\alpha_m}} = Y^p = T$. A questo punto, tenendo conto del fatto che le matrici $V^{(k)}$ e $W^{(k)}$ hanno la stessa struttura a blocchi di T, vogliamo ricavare una relazione ricorsiva simile a (2.5), che ci permetta di calcolare i blocchi della matrice Y in tempo logaritmico rispetto a p. Analogamente a quanto abbiamo visto nel paragrafo precedente

$$V_{ij}^{(k+1)} = \left(V^{(k)} V^{(k)}\right)_{ij} = V_{ii}^{(k)} V_{ij}^{(k)} + V_{ij}^{(k)} V_{jj}^{(k)} + B_{ij}^{(k)} \tag{2.7}$$

con $B_{ij}^{(k)} = \sum_{h=i+1}^{j-1} V_{ih}^{(k)} V_{hj}^{(k)}$ e

$$W_{ij}^{(k+1)} = W_{ii}^{(k)} V_{ij}^{(\alpha_k+2)} + W_{ij}^{(k)} V_{jj}^{(\alpha_k+2)} + C_{ij}^{(k)} \tag{2.8}$$

con $C_{ij}^{(k)} = \sum_{h=i+1}^{j-1} W_{ih}^{(k)} V_{hj}^{(\alpha_k+2)}$.

Iniziamo considerando il caso $p = 2^\alpha$. Per come abbiamo definito le matrici $V^{(k)}$ abbiamo $T_{ij} = Y_{ij}^{2^\alpha} = V_{ij}^{(\alpha+2)}$, usando la formula (2.7) e sostituendo ricorsivamente possiamo ricavare un'equazione che coinvolga, oltre ai blocchi diagonali, solo $V^{(2)}$ e $V^{(\alpha+2)}$.

Teorema 2.1. Siano $p = 2^\alpha$, T quasi triangolare superiore con blocchi T_{ij} per $i, j = 1 : s$, Y radice p-esima primaria di T, $V^{(k)}$ date da (2.6) allora, $\forall i < j$

$$T_{ij} = \sum_{h=0}^{\alpha} \sum_{l=0}^{2^{\alpha-h}-1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{2^\alpha - 2^h(l+1)} \tag{2.9}$$

dove $B_{ij}^{(1)} = Y_{ij}$ e $B_{ij}^{(h)} = \sum_{k=i+1}^{j-1} V_{ik}^{(h)} V_{kj}^{(h)}$

Dimostrazione. Procediamo per induzione su α.

Per $\alpha = 0$ in (2.9) abbiamo solo il termine con $h = l = 0$, quindi la tesi equivale a

$$T_{ij} = Y^0_{ii} B^{(0)}_{ij} Y^0_{jj} = B^{(0)}_{ij} = Y_{ij}$$

che è vero perché $Y^p_{ij} = T_{ij}$ e $p = 2^\alpha = 1$.

Supponiamo che la tesi sia verificata per $p = 2^\alpha$ e dimostriamola per $p = 2^{\alpha+1}$.

$$T_{ij} = Y^{2^{\alpha+1}}_{ij} = V^{(\alpha+3)}_{ij} = V^{(\alpha+2)}_{ii} V^{(\alpha+2)}_{ij} + V^{(\alpha+2)}_{ij} V^{(\alpha+2)}_{jj} + B^{(\alpha+2)}_{ij} \tag{2.10}$$

Per definizione abbiamo $V^{(\alpha+2)}_{ii} = Y^{2^\alpha}_{ii}$, mentre per ipotesi induttiva

$$V^{(\alpha+2)}_{ij} = Y^{2^\alpha}_{ij} = \sum_{h=0}^{\alpha} \sum_{l=0}^{2^{\alpha-h}-1} Y^{2^h l}_{ii} B^{(h+1)}_{ij} Y^{2^\alpha - 2^h(l+1)}_{jj}$$

e sostituendo in (2.10) abbiamo

$$\begin{aligned} T_{ij} = Y^{2^\alpha}_{ii} &\left(\sum_{h=0}^{\alpha} \sum_{l=0}^{2^{\alpha-h}-1} Y^{2^h l}_{ii} B^{(h+1)}_{ij} Y^{2^\alpha - 2^h(l+1)}_{jj} \right) + \\ &+ \left(\sum_{h=0}^{\alpha} \sum_{l=0}^{2^{\alpha-h}-1} Y^{2^h l}_{ii} B^{(h+1)}_{ij} Y^{2^\alpha - 2^h(l+1)}_{jj} \right) Y^{2^\alpha}_{jj} + B^{(\alpha+2)}_{ij} = B^{(\alpha+2)}_{ij} + \\ &+ \sum_{h=0}^{\alpha} \left(\sum_{l=0}^{2^{\alpha-h}-1} Y^{2^\alpha + 2^h l}_{ii} B^{(h+1)}_{ij} Y^{2^\alpha - 2^h(l+1)}_{jj} + \sum_{l=0}^{2^{\alpha-h}-1} Y^{2^h l}_{ii} B^{(h+1)}_{ij} Y^{2^{\alpha+1} - 2^h(l+1)}_{jj} \right) \end{aligned}$$

A questo punto riscriviamo $B^{(\alpha+2)}_{ij}$

$$B^{(\alpha+2)}_{ij} = Y^0_{ii} B^{(\alpha+2)}_{ij} Y^0_{jj} = \sum_{l=0}^{2^{\alpha-(\alpha+1)+1}} Y^{2^{\alpha+1} l}_{ii} B^{(\alpha+2)}_{ij} Y^{2^{\alpha+1} - 2^{\alpha+1}(l+1)}_{jj}$$

e la prima delle due sommatorie interne

$$\sum_{l=0}^{2^{\alpha-h}-1} Y^{2^\alpha + 2^h l}_{ii} B^{(h+1)}_{ij} Y^{2^\alpha - 2^h(l+1)}_{jj} = \sum_{l=2^{\alpha-h}}^{2^{\alpha-h+1}-1} Y^{2^h l}_{ii} B^{(h+1)}_{ij} Y^{2^{\alpha+1} - 2^h(l+1)}_{jj}$$

e riprendendo l'equazione precedente otteniamo

$$
\begin{aligned}
T_{ij} = & \sum_{l=0}^{2^{\alpha-(\alpha+1)+1}} Y_{ii}^{2^{\alpha+1}l} B_{ij}^{(\alpha+2)} Y_{jj}^{2^{\alpha+1}-2^{\alpha+1}(l+1)} + \\
& + \sum_{h=0}^{\alpha} \left(\sum_{l=2^{\alpha-h}}^{2^{\alpha-h+1}-1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{2^{s+1}-2^h(l+1)} + \sum_{l=0}^{2^{\alpha-h}-1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{2^{\alpha+1}-2^h(l+1)} \right) = \\
& = \sum_{l=0}^{2^{\alpha-(\alpha+1)+1}} Y_{ii}^{2^{\alpha+1}l} B_{ij}^{(\alpha+2)} Y_{jj}^{2^{\alpha+1}-2^{\alpha+1}(l+1)} + \sum_{h=0}^{\alpha} \sum_{l=0}^{2^{\alpha-h+1}-1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{2^{\alpha+1}-2^h(l+1)} = \\
& = \sum_{h=0}^{\alpha+1} \sum_{l=0}^{2^{\alpha-h+1}-1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{2^{\alpha+1}-2^h(l+1)}
\end{aligned}
$$

che è proprio la tesi nel caso $p = 2^{\alpha+1}$.

□

Vediamo ora come ottenere un risultato analogo nel caso generale.

Teorema 2.2. Siano $p = \sum_{k=1}^{m} 2^{\alpha_k}$, T quasi triangolare superiore con blocchi T_{ij} per $i, j = 1 : s$, Y radice p-esima primaria di T, $V^{(k)}$ e $W^{(k)}$ e date da (2.6), allora, $\forall i < j$

$$
T_{ij} = \sum_{h=0}^{\lfloor \log_2 p \rfloor} \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)} + \sum_{h=2}^{m} C_{ij}^{(h)} Y_{jj}^{p-2^{\alpha_h} \lfloor \frac{p}{2^{\alpha_h}} \rfloor} \tag{2.11}
$$

dove $B_{ij}^{(1)} = Y_{ij}$, $B_{ij}^{(h)} = \sum_{k=i+1}^{j-1} V_{ik}^{(h)} V_{kj}^{(h)}$, e $C_{ij}^{(h)} = \sum_{k=i+1}^{j-1} W_{ik}^{(h)} V_{kj}^{(\alpha_h+2)}$

Dimostrazione. Dimostriamo il teorema per induzione su m.

Per $m = 1$ abbiamo $p = 2^\alpha$ e la tesi equivale a

$$
T_{ij} = \sum_{h=0}^{\lfloor 2^\alpha \rfloor} \sum_{l=0}^{\lfloor \frac{2^\alpha}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{2^\alpha - 2^h(l+1)} = \sum_{h=0}^{\alpha} \sum_{l=0}^{2^{\alpha-h}-1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{2^\alpha - 2^h(l+1)}
$$

che è vero per il teorema precedente.
Supponiamo che la tesi sia verificata per ogni k con $1 \le k \le m$ e dimostriamola per $k = m + 1$.

$$
\begin{aligned}
T_{ij} = Y_{ij}^{2^{\alpha_1} + \cdots + 2^{\alpha_{m+1}}} = W_{ij}^{(m+2)} = \left(W^{(m+1)} V^{\alpha_{m+1}+2} \right)_{ij} = \\
= W_{ii}^{(m+1)} V_{ij}^{(\alpha_{m+1}+2)} + W_{ij}^{m+1} V_{jj}^{(\alpha_{m+1}+2)} + C_{ij}^{(m+1)}
\end{aligned} \tag{2.12}
$$

Per ipotesi induttiva la tesi è vera per $p' = p - 2^{\alpha_{m+1}} = \sum_{k=0}^{m} 2^{\alpha_k}$ e $p'' = 2^{\alpha_{m+1}}$, quindi:

$$
\begin{aligned}
V_{ij}^{(\alpha_{m+1}+2)} = & \sum_{h=0}^{\lfloor \log_2 p'' \rfloor} \sum_{l=0}^{\lfloor \frac{p''}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p''-2^h(l+1)} + \sum_{h=2}^{m} C_{ij}^{(h)} Y_{jj}^{p''-2^{\alpha_h} \lfloor \frac{p''}{2^{\alpha_h}} \rfloor} = \\
& = \sum_{h=0}^{\alpha_{m+1}} \sum_{l=0}^{2^{\alpha_{m+1}-h}-1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{2^{\alpha_{m+1}}-2^h(l+1)}
\end{aligned} \tag{2.13}
$$

$$W_{ij}^{m+1} = \sum_{h=0}^{\lfloor \log_2 p' \rfloor} \sum_{l=0}^{\lfloor \frac{p'}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p'-2^h(l+1)} + \sum_{h=2}^{m} C_{ij}^{(h)} Y_{jj}^{p'-2^{\alpha_h}\lfloor \frac{p'}{2^{\alpha_h}} \rfloor} \tag{2.14}$$

Usando il fatto che $p' = p - 2^{\alpha_{m+1}}$ abbiamo

$$p' - 2^{\alpha_h}\lfloor \frac{p'}{2^{\alpha_h}} \rfloor = p - 2^{\alpha_{m+1}} - 2^{\alpha_h}\lfloor \frac{p - 2^{\alpha_{m+1}}}{2^{\alpha_h}} \rfloor = \sum_{k=h+1}^{m} 2^{\alpha_k}$$

e quindi

$$Y_{jj}^{p'-2^{\alpha_h}\lfloor \frac{p'}{2^{\alpha_h}} \rfloor} = \prod_{k=h+1}^{m} Y_{jj}^{2^{\alpha_k}} \tag{2.15}$$

sostituendo (2.15) in (2.14) e poiché $\lfloor \log_2 p' \rfloor = \alpha_1$, otteniamo

$$W_{ij}^{m+1} = \sum_{h=0}^{\alpha_1} \sum_{l=0}^{\lfloor \frac{p'}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p'-2^h(l+1)} + \sum_{h=2}^{m} \left(C_{ij}^{(h)} \prod_{l=h+1}^{m} Y_{jj}^{2^{\alpha_l}} \right). \tag{2.16}$$

A questo punto, sostituendo (2.16) e (2.13) in (2.12), ricordando che $p' = p - 2^{\alpha_{m+1}}$, $W_{ii}^{m+1} = Y_{ii}^{p-2^{\alpha_{m+1}}}$ e $V_{jj}^{\alpha_{m+1}+2} = Y_{jj}^{2^{\alpha_{m+1}}}$ otteniamo:

$$\begin{aligned} T_{ij} = C_{ij}^{(m+1)} &+ \sum_{h=0}^{\alpha_{m+1}} \sum_{l=0}^{2^{\alpha_{m+1}-h}-1} Y_{ii}^{p-2^{\alpha_{m+1}}+2^h l} B_{ij}^{(h+1)} Y_{jj}^{2^{\alpha_{m+1}}-2^h(l+1)} + \\ &+ \sum_{h=0}^{\alpha_1} \sum_{l=0}^{\lfloor \frac{p-2^{\alpha_{m+1}}}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)} + \sum_{h=2}^{m} \left(C_{ij}^{(h)} \prod_{l=h+1}^{m+1} Y_{jj}^{2^{\alpha_l}} \right). \end{aligned}$$

Mettendo insieme il primo e l'ultimo termine e spezzando il terzo otteniamo:

$$\begin{aligned} T_{ij} = &\sum_{h=0}^{\alpha_{m+1}} \sum_{l=0}^{2^{\alpha_{m+1}-h}-1} Y_{ii}^{p-2^{\alpha_{m+1}}+2^h l} B_{ij}^{(h+1)} Y_{jj}^{2^{\alpha_{m+1}}-2^h(l+1)} + \sum_{h=2}^{m+1} \left(C_{ij}^{(h)} \prod_{l=h+1}^{m+1} Y_{jj}^{2^{\alpha_l}} \right) + \\ &+ \sum_{h=0}^{\alpha_{m+1}} \sum_{l=0}^{\lfloor \frac{p-2^{\alpha_{m+1}}}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)} + \sum_{h=\alpha_{m+1}}^{\alpha_1} \sum_{l=0}^{\lfloor \frac{p-2^{\alpha_{m+1}}}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)}. \end{aligned}$$

Ponendo $k = l + \lfloor \frac{p}{2^h} \rfloor - 2^{\alpha_{m+1}-h}$ nel secondo addendo, quindi constatando che $\lfloor \frac{p-2^{\alpha_{m+1}}}{2^h} \rfloor = \lfloor \frac{p}{2^h} \rfloor$ per $h = \alpha_{m+1}+1, \ldots, \alpha_1$ e $2^h \lfloor \frac{p}{2^h} \rfloor = p$ per $h = 0, \ldots, \alpha_{m+1}$, T_{ij} risulta uguale a:

$$\begin{aligned} &\sum_{h=0}^{\alpha_{m+1}} \sum_{k=\lfloor \frac{p}{2^h} \rfloor - 2^{\alpha_{m+1}-h}}^{\lfloor \frac{p}{2^h} \rfloor - 1} Y_{ii}^{p-2^h\lfloor \frac{p}{2^h} \rfloor + 2^h k} B_{ij}^{(h+1)} Y_{jj}^{2^h \lfloor \frac{p}{2^h} \rfloor - 2^h(k+1)} + \sum_{h=2}^{m+1} \left(C_{ij}^{(h)} \prod_{l=h+1}^{m+1} Y_{jj}^{2^{\alpha_l}} \right) + \\ &+ \sum_{h=0}^{\alpha_{m+1}} \sum_{l=0}^{\lfloor \frac{p-2^{\alpha_{m+1}}}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)} + \sum_{h=\alpha_{m+1}}^{\alpha_1} \sum_{l=0}^{\lfloor \frac{p-2^{\alpha_{m+1}}}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)} = \end{aligned}$$

$$= \sum_{h=0}^{\alpha_{m+1}} \sum_{k=\lfloor \frac{p}{2^h} \rfloor - 2^{\alpha_{m+1}-h}}^{\lfloor \frac{p}{2^h} \rfloor - 1} Y_{ii}^{2^h k} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(k+1)} + \sum_{h=2}^{m+1} \left(C_{ij}^{(h)} \prod_{l=h+1}^{m+1} Y_{jj}^{2^{\alpha_l}} \right) +$$

$$+ \sum_{h=0}^{\alpha_{m+1}} \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 2^{\alpha_{m+1}} - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)} + \sum_{h=\alpha_{m+1}}^{\alpha_1} \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)}.$$

Mettendo nuovamente insieme il primo e il terzo addendo possiamo riscrivere l'ultima quantità come:

$$\sum_{h=2}^{m+1} \left(C_{ij}^{(h)} \prod_{l=h+1}^{m+1} Y_{jj}^{2^{\alpha_l}} \right) + \sum_{h=0}^{\alpha_{m+1}} \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)} +$$

$$+ \sum_{h=\alpha_{m+1}}^{\alpha_1} \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)} =$$

$$= \sum_{h=2}^{m+1} \left(C_{ij}^{(h)} \prod_{l=h+1}^{m+1} Y_{jj}^{2^{\alpha_l}} \right) + \sum_{h=0}^{\alpha_1} \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)}$$

che è quello che volevamo dimostrare.

□

Ricordando che il nostro scopo è calcolare il blocco Y_{ij}, vediamo adesso come ridurre l'equazione (2.11) ad un sistema lineare. Poiché $B_{ij}^{(1)} = Y_{ij}$ isolando il termine con $h = 0$ della prima sommatoria e mettendo in evidenza Y_{ij} otteniamo

$$\sum_{l=0}^{p-1} Y_{ii}^l Y_{ij} Y_{jj}^{p-l-1} = T_{ij} - \sum_{h=1}^{\lfloor \log_2 p \rfloor} \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)} - \sum_{h=2}^{m} C_{ij}^{(h)} Y_{jj}^{p-2^{\alpha_h} \lfloor \frac{p}{2^{\alpha_h}} \rfloor}$$

da cui

$$\sum_{l=0}^{p-1} \operatorname{vec}\left(Y_{ii}^l Y_{ij} Y_{jj}^{p-l-1} \right) = \operatorname{vec}\left(T_{ij} - \sum_{h=1}^{\lfloor \log_2 p \rfloor} \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)} - \sum_{h=2}^{m} C_{ij}^{(h)} Y_{jj}^{p-2^{\alpha_h} \lfloor \frac{p}{2^{\alpha_h}} \rfloor} \right) =$$

$$= \operatorname{vec}(T_{ij}) - \sum_{h=1}^{\lfloor \log_2 p \rfloor} \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} \operatorname{vec}\left(Y_{ii}^{2^h l} B_{ij}^{(h+1)} Y_{jj}^{p-2^h(l+1)} \right) - \operatorname{vec}\left(\sum_{h=2}^{m} C_{ij}^{(h)} \prod_{l=h+1}^{m} Y_{jj}^{2^{\alpha_l}} \right).$$

Poiché $\operatorname{vec}(AMB) = (B^T \otimes A) \operatorname{vec}(M)$

$$\begin{aligned} &\left(\sum_{l=0}^{p-1} (Y_{jj}^{p-l-1})^T \otimes Y_{ii}^l \right) \operatorname{vec}(Y_{ij}) = \operatorname{vec}(T_{ij}) - \operatorname{vec}\left(\sum_{h=2}^{m} C_{ij}^{(h)} \prod_{l=h+1}^{m} Y_{jj}^{2^{\alpha_l}} \right) + \\ &- \sum_{h=1}^{\lfloor \log_2 p \rfloor} \left(\sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} (Y_{jj}^{p-2^h(l+1)})^T \otimes Y_{ii}^{2^h l} \right) \operatorname{vec}\left(B_{ij}^{(h+1)} \right) \end{aligned} \tag{2.17}$$

Useremo questo sistema lineare per calcolare ricorsivamente tutti i blocchi della matrice Y. La differenza rispetto ai due algoritmi precedenti consiste proprio nel fatto che siamo in grado di calcolare il lato destro dell'uguaglianza e la matrice dei coefficienti con $O(\log_2 p)$ operazioni aritmetiche. Per brevità, definiamo

$$M_{ij}^{(h)} := \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} (Y_{jj}^T)^{p-2^h(l+1)} \otimes Y_{ii}^{2^h l} \qquad U_{jj}^{(h)} = \prod_{l=h}^{m} Y_{jj}^{2^{\alpha_l}}$$

e riscriviamo (2.17) con questa nuova notazione

$$M_{ij}^{(0)} \operatorname{vec}(Y_{ij}) = \operatorname{vec}(T_{ij}) - \operatorname{vec}\left(\sum_{h=2}^{m} C_{ij}^{(h)} U_{jj}^{(h+1)}\right) - \sum_{h=1}^{\lfloor \log_2 p \rfloor} M_{ij}^{(h)} \operatorname{vec}\left(B_{ij}^{(h+1)}\right) \tag{2.18}$$

Algoritmo

Vediamo adesso l'algoritmo principale del metodo di Iannazzo-Manasse.

Algoritmo 1: data T quasi triangolare superiore con blocchi T_{ij} per $i, j = 1, \ldots, s$, $f(z)$ radice p-esima primaria, calcola $f(T)$

for j=1:s **do**
  $V_{jj}^{(1)} = I$
  calcola $V_{jj}^{(2)} = f(T_{jj})$
  for h=3:t+1 **do**
    calcola $V_{jj}^{(h)} = (V_{jj}^{(h-1)})^2$
  end for
  $W_{jj}^{(1)} = I$, $W_{jj}^{(2)} = V_{jj}^{(t+1)}$, $W_{jj}^{(m+1)} = T_{jj}$
  for h=3:m **do**
    calcola $W_{jj}^{(h)} = W_{jj}^{(h-1)} V_{jj}^{\alpha_{h-1}+2}$
  end for
  $U_{jj}^{(m+1)} = I$, $U_{jj}^{(m)} = V_{jj}^{(\alpha_m+1)}$, $U_{jj}^{(1)} = T_{jj}$
  for h=m-1:2 **do**
    calcola $U_{jj}^{(h)} = U_{jj}^{(h+1)} V_{jj}^{\alpha_h+2}$
  end for
  for i=j-1:1 **do**
    for h=2:t **do**
      calcola $B_{ij}^{(h)} = \sum_{k=i+1}^{j-1} V_{ik}^{(h)} V_{kj}^{(h)}$
    end for
    for h=2:m **do**
      calcola $C_{ij}^{(h)} = \sum_{k=i+1}^{j-1} W_{ik}^{(h)} V_{kj}^{(c_h+2)}$
    end for
    for h=0:t-1 **do**

23: calcola $M_{ij}^{(h)}$ usando l'algoritmo 3
24: **end for**
25: calcola il lato destro del sistema (2.17)
26: risolvi il sistema (2.17) per Y_{ij}
27: **for** h=1:t+1 **do**
28: calcola $V_{ij}^{(h)} = V_{ii}^{(h-1)} V_{ij}^{(h-1)} + V_{ij}^{(h-1)} V_{jj}^{(h-1)} + B_{ij}^{(h-1)}$
29: **end for**
30: **for** h=1:m **do**
31: calcola $W_{ij}^{(h)} = W_{ii}^{(h-1)} V_{ij}^{(\alpha_{h-1}+2)} + V_{ii}^{(h-1)} W_{ij}^{(\alpha_{h-1}+2)} + C_{ij}^{(h)}$
32: **end for**
33: **end for**
34: **end for**

Analisi del costo

Il calcolo dei blocchi diagonali di ognuna delle matrici V, W e U (ln:2-14) richiede tempo $O(t+2m-2)$ per ogni j indice di colonna fissato. Al variare di j e di $i <= j-1$ il calcolo del blocco in posizione (i,j) richiede tempo complessivo di $O((t+m-2)(j-i-1))$ per le matrici delle successioni B e C (ln:16-21) e $O(t+m)$ per le matrici V e W (ln:27-32). Vedremo in seguito l'algoritmo 3 per il calcolo dei coefficienti del sistema (2.17), che ci permetterà di calcolare le matrici M e risolvere il sistema lineare relativo al blocco (i,j) in tempo $O(t+m)$. Al crescere di p e n la parte principale del costo totale del metodo di Iannazzo-Manasse è data quindi da $O(n^3(t+m-2)) \leq O(n^3 log_2 p)$.
Ricordiamo che stiamo supponendo che la matrice T sia già quasi triangolare superiore: per A matrice qualsiasi dobbiamo considerare il costo aggiuntivo dato dal calcolo della forma normale di Schur.

Algoritmo per il calcolo dei coefficienti

Fissata la coppia (i,j) con $i < j$, vediamo adesso un algortimo per calcolare simultaneamente le matrici $M_{ij}^{(h)}$ per $h = 0 : \lfloor \log_2 p \rfloor$.
Iniziamo considerando il caso scalare, in cui

$$m_h := \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} x^{p-2^h(l+1)} y^{2^h l} \quad \text{per } x, y \in \mathbb{C}$$

e dimostriamo un risultato che ci servirà per strutturare l'algoritmo.

Teorema 2.3. Siano $x, y \in \mathbb{C}$ e p con sviluppo binario $\sum_{j=0}^{t} b_j 2^{t-j}$, dove $t = \lfloor \log_2 p \rfloor + 1$. Definiamo $\sigma_k := x^{2^{k-1}} + y^{2^{k-1}}$ per $k = 1 : t$, $\tilde{u}_k := x^{p-2^k \lfloor \frac{p}{2^k} \rfloor}$ e $\tilde{w}_k := x^{2^{k+1} \lfloor \frac{p}{2^{k+1}} \rfloor}$ per $k = 0 : t-1$. Allora:

1. $m_h = m_{h+1}\sigma_{h+1} + b_{t-h}\tilde{u}_h\tilde{w}_h$ per $h = 0 : t-2$
2. $m_{t-1} = b_1\tilde{u}_{t-1}\tilde{w}_{t-1} = \tilde{u}_{t-1} = x^{p-2^{t-1}}$

Dimostrazione. Il punto 2. si verifica direttamente.
Poiché $\lfloor \frac{p}{2^{t-1}} \rfloor = 1$ e $\lfloor \frac{p}{2^t} \rfloor = 0$, abbiamo

$$\begin{aligned}
&\tilde{u}_{t-1} = x^{p-2^{t-1}\lfloor \frac{p}{2^{t-1}} \rfloor} = x^{p-2^{t-1}} \\
&b_1\tilde{u}_{t-1}\tilde{w}_{t-1} = 1 \cdot x^{p-2^{t-1}} y^{2^t \lfloor \frac{p}{2^t} \rfloor} = x^{p-2^{t-1}} \\
&m_{t-1} = \sum_{l=0}^{\lfloor \frac{p}{2^{t-1}} \rfloor - 1} x^{p-2^{t-1}(l+1)} y^{2^{t-1}l} = x^{p-2^{t-1}}
\end{aligned}$$

e quindi la tesi.
Per dimostrare il punto 1. abbiamo bisogno della seguente decomposizione, valida per una generica successione $\{a_l\}_l$ per $0 \le h \le t$. Poiché $\lfloor \frac{p}{2^h} \rfloor = 2\lfloor \frac{p}{2^{h+1}} \rfloor + b_{t-h}$ abbiamo

$$\begin{aligned}
\sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} a_l &= \sum_{l=0}^{2\lfloor \frac{p}{2^{h+1}} \rfloor + b_{t-h} - 1} a_l = \sum_{l=0}^{2\lfloor \frac{p}{2^{h+1}} \rfloor - 1} a_l + \sum_{l=2\lfloor \frac{p}{2^{h+1}} \rfloor}^{2\lfloor \frac{p}{2^{h+1}} \rfloor + b_{t-h} - 1} a_l = \\
&= \sum_{l=0}^{\lfloor \frac{p}{2^{h+1}} \rfloor - 1} (a_{2l} + a_{2l+1}) + \sum_{l=2\lfloor \frac{p}{2^{h+1}} \rfloor}^{2\lfloor \frac{p}{2^{h+1}} \rfloor + b_{t-h} - 1} a_l
\end{aligned}$$

Se $b_{t-h} = 1$ l'ultima sommatoria è uguale a $a_{2\lfloor \frac{p}{2^{h+1}} \rfloor} = a_{\lfloor \frac{p}{2^h} \rfloor - 1}$, mentre se $b_{t-h} = 0$ anche la sommatoria è nulla, quindi

$$\sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} a_l = \sum_{l=0}^{\lfloor \frac{p}{2^{h+1}} \rfloor - 1} (a_{2l} + a_{2l+1}) + b_{t-h} a_{\lfloor \frac{p}{2^h} \rfloor - 1}. \tag{2.19}$$

Usando questa identità abbiamo

$$\begin{aligned}
m_h &= \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} x^{p-2^h(l+1)} y^{2^h l} = \\
&= \sum_{l=0}^{\lfloor \frac{p}{2^{h+1}} \rfloor - 1} \left(x^{p-2^h(2l+1)} y^{2^h 2l} + x^{p-2^h(2l+2)} y^{2^h(2l+1)} \right) + b_{t-h} x^{p-2^h \lfloor \frac{p}{2^h} \rfloor} y^{2^h(\lfloor \frac{p}{2^h} \rfloor - 1)} = \qquad (2.20) \\
&= (x^{2^h} + y^{2^h}) \sum_{l=0}^{\lfloor \frac{p}{2^{h+1}} \rfloor - 1} (x^{p-2^{h+1}(l+1)} y^{2^{h+1}k}) + b_{t-h}\tilde{u}_h\tilde{w}_h = \sigma_{h+1} m_{h+1} + b_{t-h}\tilde{u}_h\tilde{w}_h.
\end{aligned}$$

che è la tesi.

□

Il seguente algoritmo sfrutta il teorema precedente per calcolare m_h, per $h = 0 : t-1$ con $O(\log_2 p)$ operazioni.

Algoritmo 2: dati $x, y \in \mathbb{C}$ e $p \geq 1$ calcola m_h per $h = 0, \ldots, t-1$

1: $x_1 = x$, $y_1 = y$
2: **for** h=2:t **do**
3: calcola $x_h = x_{h-1}^2$, $y_h = y_{h-1}^2$, $\sigma_h = x_h + y_h$
4: **end for**
5: $\tilde{u}_0 = 1$, $\tilde{w}_{t-1} = 1$
6: **for** h=1:t-1 **do**
7: calcola $\tilde{u}_h = \tilde{u}_{h-1} x_h^{b_{t-h1}}$
8: **end for**
9: **for** h=t-2:0 **do**
10: calcola $\tilde{w}_h = \tilde{w}_{h+1} y_{h+2}^{b_{t-h-1}}$
11: **end for**
12: $m_{t-1} = \tilde{u}_{t-1}$
13: **for** h=t-2:0 **do**
14: calcola m_h usando il Teorema 2.3
15: **end for**

Vogliamo adesso estendere l'algoritmo al caso matriciale. Siano $X \in \mathbb{C}^{n\times n}$ e $Y \in \mathbb{C}^{m\times m}$, definiamo $M_h := \sum_{l=0}^{\lfloor \frac{p}{2^h} \rfloor - 1} X^{p-2^h(l+1)} \otimes Y^{2^h l}$, abbiamo il seguente risultato:

Teorema 2.4. Date $X \in \mathbb{C}^{n\times n}$ e $Y \in \mathbb{C}^{m\times m}$, e p con sviluppo binario $\sum_{j=1}^{t} b_j 2^{t-j}$, dove $t = \lfloor \log_2 p \rfloor + 1$. Definiamo $S_k := X^{2^{k-1}} \otimes I_m + I_n \otimes y^{2^{k-1}}$ per $k = 1 : t$, $\tilde{U}_k := X^{p-2^k \lfloor \frac{p}{2^k} \rfloor}$ e $\tilde{W}_k := X^{2^{k+1} \lfloor \frac{p}{2^{k+1}} \rfloor}$ per $k = 0 : t-1$. Allora:

1. $M_h = M_{h+1} S_{h+1} + b_{t-h} \tilde{U}_h \otimes \tilde{W}_h$ per $h = 0 : t-2$
2. $M_{t-1} = X^{p-2^{t-1}} \otimes I_m$

Dimostrazione. La dimostrazione ricalca esattamente gli stessi passi del Teorema 2.3, la vedremo quindi molto brevemente.
Il punto 2. si verifica direttamente. Per il punto 1., adattando la decomposizione (2.20) e sostituendo $\tilde{U}_h$ e $\tilde{W}_h$ abbiamo

$$M_h = \sum_{l=0}^{\lfloor \frac{p}{2^{h+1}} \rfloor - 1} \left(X^{p-2^h(2l+1)} \otimes Y^{2^h 2l} + X^{p-2^h(2l+2)} \otimes Y^{2^h(2l+1)} \right) + b_{t-h} \tilde{U}_h \otimes \tilde{W}_h.$$

Raccogliendo $(X^{2^h} \otimes I_m + I_n \otimes y^{2^h}) = S_{h+1}$ e ricordando che $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$, otteniamo:

$$M_h = \left(\sum_{l=0}^{\lfloor \frac{p}{2^{h+1}} \rfloor - 1} X^{p-2^{h+1}(l+1)} \otimes Y^{2^{h+1}k} \right) S_{h+1} + b_{t-h} \tilde{U}_h \tilde{W}_h = M_{h+1} S_{h+1} + b_{t-h} \tilde{U}_h \otimes \tilde{W}_h$$

□

Vediamo adesso come riadattare l'algoritmo visto per il calcolo degli scalari m_h per calcolare le matrici $M_{ij}^{(h)}$ del sistema lineare (2.17).

Algoritmo 3: calcola $M_{ij}^{(h)}$ all'interno di Algoritmo 1, dove s_k dimensione di V_{kk}

1: **for** h=0:t **do**
2: $X_h = (V_{jj}^{h+1})^T$ e $Y_h = V_{ii}^{h+1}$
3: **end for**
4: **for** h=0:t **do**
5: calcola $S_h = X_h \otimes I_{s_i} + I_{s_j} Y_h$
6: **end for**
7: $\tilde{U}_0 = I_{s_j}$
8: **for** h=1:t-1 **do**
9: $\theta = 1 + \sum_{k=1}^{t-h} b_k$
10: calcola $\tilde{U}_h = (U_{jj}^{(\theta)})^T$
11: **end for**
12: $\tilde{W}_{t-1} = I_{s_i}$
13: **for** h=0:t-2 **do**
14: $\theta = 1 + \sum_{k=1}^{t+h-1} b_k$
15: calcola $\tilde{W}_h = W_{ii}^{(\theta)}$
16: **end for**
17: calcola $M_{ij}^{(t-1)} = \tilde{U}_{t-1} \otimes I_{s_i}$
18: **for** h=t-2:0 **do**
19: calcola M_{ij}^{h} usando il Teorema 2.4
20: **end for**

Osserviamo che, poichè questo algoritmo sarà usato all'interno dell'algoritmo 1, molte delle matrici coinvolte saranno già note: gli unici passi che richiedono effettivamente operazioni aggiuntive sono il calcolo delle matrici S (ln: 4-6) e M (ln: 17-19) che necessitano complessivamente di $O(t + m)$ operazioni.

Generalizzazioni

L'algoritmo visto si può applicare anche nel caso in cui la matrice T sia triangolare a blocchi con blocchi diagonali di dimensione arbitraria. L'unico problema che si presenta è che, mentre $f(T_{ii})$ con T_{ii} blocco diagonale di dimensione 2×2 può essere calcolato direttamente, quando la dimensione del blocco è superiore non abbiamo alcuna formula esplicita.
Nel caso in cui la matrice A sia complessa la generalizzazione è immediata: la forma normale di Schur sarà data da una matrice T triangolare superiore a cui possiamo applicare l'algoritmo che abbiamo descritto per le matrici triangolari a blocchi, ricordando che avremo tutti blocchi di ordine 1.

Potenze arbitrarie

Consideriamo ora il caso in cui vogliamo calcolare la potenza $\frac{q}{p}$-esima di una matrice A, con p e q interi coprimi. Il problema potrebbe essere risolto banalmente calcolando la radice p-esima di A^q oppure calcolando la radice p-esima di A elevandola successivamente alla q: per ottenere un algoritmo efficiente utilizzeremo questa seconda opzione, con alcuni accorgimenti per quanto riguarda l'elevamento a potenza.
Sia f la funzione *potenza* $\frac{q}{p}$*-esima* e $Z := f(A)$ la matrice che vogliamo calcolare. Presa la forma di normale di Schur di $A = QTQ^H$, calcoliamo la matrice Y radice p-esima di T usando l'algoritmo 1, ottenendo $X := QYQ^H$ radice p-esima di A. A questo punto calcoliamo la $q-$esima potenza di Y, usando il *binary powering*: sia $q = \sum_{k=1}^{r} 2^{d_k}$ con $d_1 > d_2 > \cdots > d_r \geq 0$, siano $V^{(k)}$ le matrici definite in (2.6), $Z^{(k)} := Z^{(k-1)}V^{(d_k+2)}$ per k=2:r e $Z^{(1)} := V^{d_1+2} = Y^{2^{d_1}}$, possiamo calcolare $Y^q = Z^{(r)}$ sfruttando queste relazioni di ricorrenza e il fatto che le matrici $V^{(k)}$ sono già state calcolate durante l'esecuzione dell'algoritmo 1 per la radice.

2.2 Algoritmo di Schur-Padé

Nei prossimi paragrafi descriveremo un algoritmo per il calcolo di potenze arbitrarie di matrici basato sulle approssimanti di Padé. Inizieremo quindi con alcuni risultati generali su tale approssimazione, per poi passare alla derivazione dell'algoritmo. Faremo riferimento all'orticolo originale di Higham [6].

2.2.1 Approssimanti di Padé

Per tutte le definizioni e i risultati presenti in questo paragrafo facciamo riferimento a Baker [1].

Definizione 2.1. Data una funzione analitica $f(x)$, chiamiamo *approssimante di Padé* $[k/m]$ la funzione razionale $\mathcal{R}_{km}(x) = \dfrac{P_{km}(x)}{Q_{km}(x)}$ con P e Q polinomi coprimi, $\deg(P_{km}) \leq k$, $\deg(Q_{km}) \leq m$ e $Q_{km}(0) = 1$, tale che

$$f(x) - \mathcal{R}_{km}(x) = O(x^{k+m+1}). \tag{2.21}$$

La condizione (2.21) induce la connessione tra l'approssimante di Padé e i primi $k+m+1$ termini dello sviluppo di Taylor di f. In particolare, l'approssimante $\mathcal{R}_{k0}$ sarà proprio il polinomio dato dal troncamento della serie di Taylor al $k-$esimo addendo.
Vediamo adesso alcuni risultati riguardo esistenza e unicità di tali approssimanti.

Teorema 2.5. Data $f(x)$ che ammette sviluppo di Taylor in 0 con $f(0) \neq 0$ e $k, m \in \mathbb{N}$, se f ammette un'approssimante di Padé $[k/m]$, tale approssimante è unica.

Dimostrazione.
Supponiamo $\mathcal{R}_{km}(x) = \frac{P_{km}(x)}{Q_{km}(x)}$ e $\mathcal{R}'_{km}(x) = \frac{P'_{km}(x)}{Q'_{km}(x)}$ approssimanti di Padé $[k/m]$. Per definizione abbiamo

$$f(x) - \mathcal{R}_{km}(x) = O(x^{k+m+1}) \quad \text{e} \quad f(x) - \mathcal{R}'_{km}(x) = O(x^{k+m+1})$$

quindi

$$\frac{P_{km}(x)}{Q_{km}(x)} - \frac{P'_{km}(x)}{Q'_{km}(x)} = O(x^{k+m+1})$$

che implica

$$Q'_{km}(x)P_{km}(x) - Q_{km}(x)P'_{km}(x) = O(x^{k+m+1})$$

Ma il polinomio a sinistra ha grado minore o uguale di $m+k$, quindi, affinchè l'uguaglianza sia verificata, tale polinomio dev'essere identicamente nullo, e questo ci dà la tesi.

□

Sia $f(x) = \sum_{j=0}^{\infty} a_j x^j$ e supponiamo che i due polinomi approssimanti siano dati da $P_{km}(x) = \sum_{j=0}^{k} p_j x^j$ e $Q_{km}(x) = 1 + \sum_{j=1}^{m} q_j x^j$, dalla condizione (2.21) abbiamo

$$Q_{km}(x)f(x) = P_{km}(x) + O(x^{k+m+1})$$

ed eguagliando i coefficienti dei monomi dello stesso grado, otteniamo il sistema lineare

$$\begin{cases} a_0 = p_0 \\ q_0 a_1 + a_1 = p_1 \\ \quad \vdots \\ q_0 a_k + \cdots + q_m a_{k-m} = p_k \end{cases} \qquad \begin{cases} q_0 a_{k+1} + \cdots + q_m a_{k+1-m} = 0 \\ \quad \vdots \\ q_0 a_m + k + \cdots + q_m a_k = 0 \end{cases}$$

che per la regola di Cramer equivale a

$$P_{km}(x) = \det \begin{pmatrix} a_{k-m+1} & a_{k-m+2} & \dots & a_{k+1} \\ \vdots & \vdots & & \vdots \\ a_k & a_{k+1} & \dots & a_{k+m} \\ \sum_{j=m}^{k} a_{j-m}x^j & \sum_{j=m-1}^{k} a_{j-m+1}x^j & \dots & \sum_{j=0}^{k} a_j x^j \end{pmatrix} \tag{2.22}$$

$$Q_{km}(x) = \det \begin{pmatrix} a_{k-m+1} & a_{k-m+2} & \dots & a_{k+1} \\ \vdots & \vdots & & \vdots \\ a_k & a_{k+1} & \dots & a_{k+m} \\ x^m & x^{m-1} & \dots & 1 \end{pmatrix} \tag{2.23}$$

ma questo ci fornisce una condizione sufficiente per l'esistenza dell'approssimante $[k/m]$: detto $C[k/m] := \det \begin{pmatrix} a_{k-m+1} & \dots & a_k \\ \vdots & & \vdots \\ a_k & \dots & a_{k+m-1} \end{pmatrix}$ da (2.23) abbiamo che il termine noto di Q_{km} è proprio $C[k/m]$, e quindi se $C[k/m] \neq 0$ allora $Q_{km} \neq 0$ e l'approssimante esiste.
Vediamo ora alcune proprietà dell'approssimante di Padé che ci serviranno nel resto del capitolo, per tutti i risultati di cui non riportiamo la dimostrazione rimandiamo a Baker [1].

Lemma 2.1. Data $f(x)$ che ammette sviluppo di Taylor in 0 con $f(0) \neq 0$, se $\mathcal{R}_{km}(x)$ approssimante di Padé $[k/m]$ di $f(x)$ allora $\dfrac{1}{\mathcal{R}_{km}(x)}$ è l'approssimante $[m/k]$ di $\dfrac{1}{f(x)}$

Definizione 2.2. $w(x)$ si dice *funzione peso* su (a, b) se verifica le seguenti proprietà:

1. $w(x)$ positiva e continua
2. $\forall n \in \mathbb{N}$ la funzione $|x^n|w(x)$ è integrabile su (a, b)
3. se g è una funzione continua e positiva su (a, b) tale che $\int_a^b g(x)w(x)dx = 0$ allora $g(x) = 0 \ \ \forall x \in (a, b)$

Si può dimostrare che per ogni w funzione peso su (a, b) esistono $\varphi_0, \varphi_1, \dots$ polinomi ortogonali tali che $\deg(\varphi_n) = n$ per ogni n e che per ogni $k \neq n$ $\int_a^b \varphi_k(x)\varphi_n(x)w(x)dx = 0$. Tali polinomi saranno detti *associati* alla funzione peso.
Il seguente risultato, di cui omettiamo la dimostrazione, mostra la connessione tra approssimanti di Padé e polinomi ortogonali.

Teorema 2.6. Data $f(x) = \sum_{n=0}^{+\infty} a_n x^n$ in un intorno di 0, sia w funzione peso su (a, b) con $0 \leq a < b$ tale che per ogni n

$$\int_a^b x^n w(x)dx = a_n$$

e siano ${}_j\psi_0, \ldots, {}_j\psi_m$ i polinomi associati alla funzione peso $\overline{w}(x) := x^j w(x)$.
L'approssimante di Padé $[k/m]$ di f ha denominatore

$$Q_{km}(x) = \begin{cases} x^m {}_j\psi_m\left(\frac{1}{x}\right) & x \neq 0 \\ 1 & x = 0 \end{cases}$$

con $j = k - m + 1$.

2.2.2 Derivazione dell'algoritmo

Data una matrice $A \in \mathbb{C}^{n\times n}$ non singolare tale che A non ha alcun autovalore reale negativo e dato un generico $p \in (-1, 1)$, vogliamo calcolare A^p radice p-esima principale della matrice A. Poiché $A^p = \exp(p \ln A)$ dove scegliamo la determinazione principale del logaritmo, potremmo usare il metodo di *scaling e squaring inverso* per calcolare $X := \ln(A)$ e successivamente il metodo di *scaling and squaring* per calcolare $\exp(pX)$. Questo procedimento richiederebbe il calcolo di due approssimanti di Padé, uno per il logaritmo e uno per l'esponenziale, costruiremo invece un algoritmo che richiederà la costruzione della sola approssimante per la funzione $(1 - x)^p$.
Per prima cosa considereremo il condizionamento di A^p, cercheremo poi un'approssimante di Padé per $(1-x)^p$ studiando la sua valutazione e l'errore di approssimazione, dopodiché presenteremo l'algoritmo vero e proprio.

Condizionamento

Ricordiamo che data una funzione scalare f e un punto x nel suo dominio, il *condizionamento relativo* di f in x è dato da

$$\mathcal{C}_f(x) := \lim_{\varepsilon\to 0} \sup_{|\Delta x|\leq\varepsilon|x|} \left|\frac{f(x+\Delta x) - f(x)}{\varepsilon f(x)}\right|$$

e, se f è differenziabile, vale $\mathcal{C}_f(x) = \left|\dfrac{xf'(x)}{f(x)}\right|$.
Analogamente, se f è una funzione di matrici e $\|\ .\ \|$ una norma indotta, abbiamo

$$\mathcal{C}_f(X) := \lim_{\varepsilon\to 0} \sup_{\|E\|\leq\varepsilon\|X\|} \frac{\|f(X+E) - f(X)\|}{\varepsilon\|f(X)\|} = \|L_f(X)\|\frac{\|X\|}{\|f(X)\|} \tag{2.24}$$

dove $L_f(X)$ è la derivata di Fréchet di f in X e $\|L_f(X)\| := \max\limits_{Z\neq 0} \dfrac{\|L_f(X,Z)\|}{\|Z\|}$.
Vediamo ora come derivare delle limitazioni per $\|L_f(X)\|$. Per il Lemma 1.1 sulla convergenza di serie di Taylor matriciali, abbiamo $(A + \varepsilon I)^p = A^p + pA^{p-1}\varepsilon + O(\varepsilon^2)$, quindi

$$L_{x^p}(A, I) = (A + I)^p - A^p + o(\|I\|) = pA^{p-1} + o(1)$$

ed abbiamo

$$\|L_{x^p}(X)\| = \max_{Z\neq 0} \frac{\|L_f(A,Z)\|}{\|Z\|} \geq \frac{\|L_f(X,I)\|}{\|I\|} = \frac{|p|\|A^{p-1}\|}{\|I\|}.$$

D'altra parte, per la regola di derivazione di funzioni composte e usando la linearità della derivata, otteniamo

$$\|L_{x^p}(A,E)\| = \|L_{e^{(p\ln x)}}(A,E)\| = \|L_{\exp}(p\ln(A), L_{p\ln}(A,E))\| = \|pL_{\exp}(p\ln(A), L_{\ln}(A,E))\| =$$

poiché la derivata di Fréchet dell'esponenziale è data da $L_{\exp}(A,E) = \int_0^1 e^{A(1-s)}Ee^{As}ds$

$$= |p| \left\| \int_0^1 e^{((1-s)p\ln A)} L_{\ln}(A,E) e^{(sp\ln A)} ds \right\| \leq |p| \int_0^1 \left\| e^{(1-s)p\ln A} L_{\ln}(A,E) e^{sp\ln A} \right\| ds \leq$$
$$\leq |p|\|L_{\ln}(A,E)\| \int_0^1 e^{(1-s)|p|\|\ln A\|} e^{s|p|\|\ln A\|} ds = |p|\|L_{\ln}(A,E)\| \int_0^1 e^{|p|\|\ln A\|} ds =$$
$$= |p|\|L_{\ln}(A,E)\| e^{|p|\|\ln A\|} \leq |p|\|L_{\ln}(A)\|\|E\| e^{|p|\|\ln A\|}$$

dove nell'ultima disuguaglianza abbiamo usato la definizione di $\|L_{\ln}(A)\|$.
In definitiva, abbiamo trovato

$$\frac{|p|\|A^{p-1}\|}{\|I\|} \leq \|L_{x^p}(A)\| \leq |p|\|L_{\ln}(A)\| e^{|p|\|\ln A\|} \tag{2.25}$$

Approssimante di Padé per $(1-x)^p$

In questa sezione vedremo come trovare un'approssimante di Padé per la funzione $(1-x)^p$ e daremo una stima per l'errore di approssimazione.
Consideriamo, per iniziare, la funzione ipergeometrica

$${}_2F_1(\alpha,\beta;\gamma;x) = \sum_{n=0}^{+\infty} \frac{b_n x^n}{n!} = \sum_{n=0}^{+\infty} \frac{(\alpha)_n(\beta)_n}{n!(\gamma)_n} x^n$$

con

$$\begin{cases} b_0 = 1 \\ b_{n+1} = \dfrac{(n+\alpha)(n+\beta)}{(n+\gamma)} b_n \end{cases} \qquad \begin{cases} (a)_0 = 1 \\ (a)_n = a(a+1)\cdot\cdots\cdot(a+n-1)\ . \end{cases}$$

Sappiamo che ${}_2F_1(\alpha,\beta;\gamma;x)$ converge per $|x| < 1$, e quindi per $X \in \mathbb{C}^{n\times n}$ la serie ${}_2F_1(\alpha,\beta;\gamma;X)$ converge per $\rho(X) < 1$, inoltre poiché $(1)_n = n!$ abbiamo

$${}_2F_1(-p,1;1;x) = \sum_{n=0}^{+\infty} \frac{(-p)_n n!}{n!n!} x^n = 1 - px + \frac{p(1-p)x^2}{2} + \cdots = (1-x)^p$$

e analogamente

$${}_2F_1(p,1;1;x) = (1-x)^{-p}$$

sfruttando questa corrispondenza dimostreremo i risultati di cui abbiamo bisogno riguardo le approssimazioni di $(1-x)^p$.

Lemma 2.2. Per ogni scelta di α, β e γ e per ogni $k, m \in \mathbb{N}$ che verificano $k-m+1 \geq 0$, esiste $\mathcal{R}_{km}$ approssimante di Padé $[k/m]$ di ${}_2F_1(\alpha, \beta; \gamma; x)$.

Lemma 2.3. Per ogni $p \in \mathbb{R}$ e per ogni $k, m \in \mathbb{N}$ esiste $\mathcal{R}_{km}$ approssimante di Padé $[k/m]$ per $(1-x)^p$.

Dimostrazione. Per il lemma 2.2, per ogni $k \geq m$ esistono $\mathcal{R}_{km}$ e $\mathcal{R}'_{km}$ approssimanti di Padé rispettivamente di ${}_2F_1(-p, 1; 1; x) = (1-x)^p$ e ${}_2F_1(p, 1; 1; x) = (1-x)^{-p}$, mentre per il lemma 2.1 l'approssimante di Padé $[m/k]$ di $(1-x)^p$ è uguale a $\dfrac{1}{\mathcal{R}'_{km}}$ e quindi abbiamo la tesi.

□

Presentiamo ora alcuni importanti risultati riguardo il polinomio $Q_{km}(x)$ denominatore dell'approssimante di Padé, facciamo riferimento a Kenney e Laub [11].

Teorema 2.7. Sia $f(x) = {}_2F_1(\alpha, 1; \gamma; x)$ con $0 < \alpha < \gamma$ e $\mathcal{R}_{km} = \frac{P_{km}(x)}{Q_{km}(x)}$ approssimante di Padé $[k/m]$ per f con $k-m+1 \geq 0$, allora:

1. tutti gli zeri di Q_{km} sono semplici e sono contenuti in $(1, +\infty)$
2. se $X \in \mathbb{C}^{n\times n}$ con $\|X\| \leq 1$, abbiamo

$$\mathcal{C}_{Q_{km}}(X) \leq \frac{Q_{km}(-\|X\|)}{Q_{km}(\|X\|)}$$

Dimostrazione.
Abbiamo bisogno del seguente risultato preliminare.

Lemma 2.4. Per $0 < \alpha < \gamma$, per ogni $n \in \mathbb{N}$ vale

$$\frac{(\alpha)_n}{(\gamma)_n} = \int_0^1 x^n w(x) dx$$

con $w(x) = \frac{\Gamma(\gamma)}{\Gamma(\alpha)\Gamma(\gamma-\alpha)} x^{\alpha-1}(1-x)^{\gamma-\alpha-1}$.

Usando il fatto che $(1)_n = n!$ e il lemma appena visto abbiamo

$${}_2F_1(\alpha, 1; \gamma; x) = \sum_{n=0}^{+\infty} \frac{(\alpha)_n}{(\gamma)_n} x^n = \sum_{n=0}^{+\infty} \left(\int_0^1 x^n w(x) dx \right) x^n$$

e quindi se $\dfrac{P_{km}(x)}{Q_{km}(x)}$ è un'approssimante di Padé $[k/m]$ per ${}_2F_1(\alpha, 1; \gamma; x)$, per il Teorema 2.6 abbiamo

$$Q_{km}(x) = x^m {}_j\psi_m \left(\frac{1}{x} \right) \tag{2.26}$$

con $j = k-m+1$ e ${}_j\psi_m$ polinomio di grado m associato all funzione peso $\overline{w}(x) = x^j w(x)$. Da questo segue immediatamente la prima parte della tesi: per (2.26) gli zeri di $Q_{km}(x)$ sono gli inversi degli zeri di ${}_j\psi_m$ che sono tutti distinti in $(0,1)$, e quindi abbiamo 1.
Dimostriamo 2. Per quanto abbiamo visto fin'ora, abbiamo

$$Q_{km}(x) = \frac{\prod_{j=1}^{m}(x_j - x)}{\prod_{j=1}^{m} x_j} \qquad \text{con } x_j > 1$$

e in particolare $Q_{km}(x)$ è una funzione decrescente e positiva per $x \leq 1$.
Se $X \in \mathbb{C}^{n\times n}$ con $\|X\| \leq 1$ allora:

$$\|Q_{km}(X)\| = \left\| \frac{\prod_{j=1}^{m}(x_j I - X)}{\prod_{j=1}^{m} x_j} \right\| \leq \frac{\prod_{j=1}^{m}(x_j + \|X\|)}{\prod_{j=1}^{m} x_j} = Q_{km}(-\|X\|). \tag{2.27}$$

Poiché per ipotesi $\|X\| \leq 1$, abbiamo

$$(x_j I - X)^{-1} = \frac{1}{x_j}\left(I - \frac{X}{x_j}\right)^{-1} = \frac{1}{x_j}\sum_{k=0}^{+\infty}\left(\frac{X}{x_j}\right)^k$$

$$\left\|(x_j I - X)^{-1}\right\| \leq \sum_{k=0}^{+\infty}\left(\frac{\|X\|}{x_j}\right)^k = \frac{1}{x_j}\left(1 - \frac{\|X\|}{x_j}\right)^{-1} = (x_j - \|X\|)^{-1}$$

e otteniamo una maggiorazione per $\|Q_{km}(X)^{-1}\|$:

$$\begin{aligned} \|Q_{km}(X)^{-1}\| &= \left\| \prod_{j=1}^{m} x_j \prod_{j=1}^{m} (x_j I - X)^{-1} \right\| \leq \prod_{j=1}^{m} x_j \prod_{j=1}^{m} \left\|(x_j I - X)^{-1}\right\| \leq \\ &\leq \prod_{j=1}^{m} x_j \prod_{j=1}^{m} (x_j - \|X\|)^{-1} = Q_{km}(\|X\|)^{-1}. \end{aligned} \tag{2.28}$$

Mettendo insieme (2.27) e (2.28) abbiamo

$$\mathcal{C}_{Q_{km}}(X) = \|Q_{km}(X)\|\|Q_{km}(X)^{-1}\| \leq \frac{Q_{km}(-\|X\|)}{Q_{km}(\|X\|)}$$

che è la tesi.

□

Corollario 2.1. Siano $p \in (-1,1)$, $\mathcal{R}_{km} = \dfrac{P_{km}(x)}{Q_{km}(x)}$ approssimante di Padé $[k/m]$ di $(1-x)^p$ con $k \geq m$, allora:

1. tutti gli zeri di Q_{km} sono semplici e sono contenuti in $(1, +\infty)$
2. se $X \in \mathbb{C}^{n\times n}$ con $\|X\| \leq 1$, abbiamo

$$\mathcal{C}_{Q_{km}}(X) \leq \frac{Q_{km}(-\|X\|)}{Q_{km}(\|X\|)}$$

Inoltre se $-1 < p < 0$ la tesi è vera per $k \geq m-1$

Dimostrazione.
Se $p \in (-1,0)$, poiché $(1-x)^p = {}_2F_1(-p,1;1;x)$, per $k \geq m-1$ siamo nelle ipotesi del teorema precedente che ci dà la tesi.
In generale, per $p \in (-1,1)$ si verifica direttamente che

$$(1-x)^p = 1 - px\,{}_2F_1(1-p,1;2;x)$$

quindi se $\mathcal{R}'_{(k-1)m} = \frac{P'_{(k-1)m}(x)}{Q'_{(k-1)m}(x)}$ è l'approssimante di Padé $[k-1/m]$ di ${}_2F_1(-p,1;2;x)$ allora $\mathcal{R}_{km} = 1 - px\mathcal{R}'_{(k-1)m}$ è l'approssimante di Padé $[k/m]$ per $(1-x)^p$ ed ha lo stesso denominatore di $\mathcal{R}'_{(k-1)m}$. Poiché se $k \geq m$ per ${}_2F_1(-p,1;2;x)$ siamo nelle ipotesi del teorema precedente, abbiamo la tesi. □

Date due approssimanti $\mathcal{R}_{km}$ e $\mathcal{R}_{k+1m-1}$, si può dimostrare (vedi Kenney e Laub [11]) che gli zeri dei due denominatori sono interlacciati, ovvero che detti $x_1,\ldots,x_m$ e $z_1,\ldots,z_{m-1}$ rispettivamente gli zeri di Q_{km} e Q_{k+1m-1}, si ha $1 < x_1 < z_1 < x_2 < \cdots < x_{m-1} < z_{m-1} < x_m$.
Definiamo l'errore di approssimazione di Padé $[k/m]$ per ${}_2F_1(\alpha,1;\gamma;x)$ come

$$E({}_2F_1(\alpha,1;\gamma;.),k,m,x) := {}_2F_1(\alpha,1;\gamma;x) - \mathcal{R}_{km}(x)$$

Vediamo adesso come ottenere un'espansione in serie della funzione di errore, per poi ricavare una limitazione superiore alla sua norma.

Teorema 2.8. Dati $k-m+1 \geq 0$ e $\alpha \notin \mathbb{Z}^-$, vale

$$E({}_2F_1(\alpha,1;\gamma;.),k,m,x) = \frac{Q_{km}(1)}{Q_{km}(x)} \sum_{n=k+m+1}^{+\infty} \frac{(\alpha)_n (n-(k+m))_m}{(\gamma)_n (n+\alpha+m)_m} x^n$$

Dimostrazione.
Consideriamo la serie

$$\sum_{n=k+m+1}^{+\infty} C_n^{(km)} x^n := Q_{km}(x)\,{}_2F_1(\alpha,1;\gamma;x) - P_{km}(x)$$

Si può dimostrare che il denominatore dell'approssimante di Padé $[k/m]$ verifica

$$Q_{km}(x) = \sum_{n=0}^{m} \frac{(-m)_n(-(\alpha+k))_n}{n!(1-(\gamma+k+m))_n} \tag{2.29}$$

usando tale risultato ed eguagliando i coefficienti dei monomi dello stesso grado otteniamo

$$\begin{aligned} C_n^{(mk)} &= \sum_{j=0}^{m} \frac{(-m)_j(-(\alpha+k))_j(\alpha)_j}{j!(1-(\gamma+k+m))_j(\gamma)_j} = \frac{(\alpha_n)}{(\gamma)_n} \sum_{j=0}^{m} \frac{(-m)_j(-(\alpha+k))_j(-(\gamma+n-1))_j}{j!(1-(\gamma+k+m))_j(-(\alpha+n-1))_j} = \\ &= \frac{(\alpha_n)}{(\gamma)_n}\,{}_3F_2(-m,-(\alpha+k),-(\gamma+n-1);1-(\gamma+k+m),-(\alpha+n-1);1) = \\ &= \frac{(\alpha_n)}{(\gamma)_n} \frac{(1+(-\gamma-m+\alpha)_m(n-(m+k)_n}{(1-(\gamma+k+m))_m(n+\alpha-m)_n} \end{aligned}$$

dove nell'ultimo passaggio abbiamo usato la formula di Saalschütz [2].
Utilizzando (2.29) si può dimostrare che $Q_{km}(1) = \dfrac{(1+(-\gamma-m+\alpha)_m}{(1-(\gamma+k+m)_m}$, sostituendo quest'uguaglianza nella precedente otteniamo

$$C_n^{(mk)} = \frac{(\alpha_n)}{(\gamma)_n}\frac{(n-(m+k))_n}{(n+\alpha-m)_n}\, Q_{km}(1).$$

Ricordando la definizione dell'errore di approssimazione e usando quanto abbiamo appena dimostrato, abbiamo la tesi:

$$E({}_2F_1(1-p,1;2;.),k,m,x) = \frac{Q_{km}(x)\,{}_2F_1(1-p,1;2;x) - P_{km}(x)}{Q_{km}(x)} =$$
$$= \frac{Q_{km}(1)}{Q_{km}(x)} \sum_{n=k+m+1}^{+\infty} \frac{(\alpha)_n(n-(k+m))_m}{(\gamma)_n(n+\alpha+m)_m}x^n$$

□

Vediamo ora come usare questo risultato per ottenere una maggiorazione in norma dell'errore di approssimazione.

Teorema 2.9. Dati $p \in (-1,1)$, $X \in \mathbb{C}^{n\times n}$ tale che $\|x\| \leq 1$, per ogni k, m interi tali che $k \geq m$ vale

$$\left\|E((1-X)^p,k,m,X)\right\| \leq \left|E((1-\|X\|)^p,k,m,\|X\|)\right|$$

Dimostrazione. Per ogni X con $\|X\| \leq 1$ vale $(I-X)^p = {}_2F_1(-p,1;1;X)$ e usando il Teorema 2.8 abbiamo

$$\left\|E((1-X)^p,k,m,X)\right\| \leq \left\|Q_{km}(1)Q_{km}(x)^{-1}\sum_{j=k+m+1}^{+\infty}\frac{(-p)_j(j-(k+m))_m}{j!(n-p+m)_m}x^j\right\| \leq$$
$$\leq |Q_{km}(1)|\left\|Q_{km}(x)^{-1}\right\|\left|\sum_{j=k+m+1}^{+\infty}\frac{(-p)_j(j-(k+m))_m}{j!(j-p+m)_m}\|X\|^j\right| \leq$$
$$\leq \left|Q_{km}(1)Q_{km}(\|X\|)^{-1}\sum_{j=k+m+1}^{+\infty}\frac{(-p)_j(j-(k+m))_m}{j!(j-p+m)_m}\|X\|^j\right|$$

dove nell'ultima disuguaglianza abbiamo usato il fatto che per $p \in (-1,1)$ e $k \geq m$ siamo nelle ipotesi del Corollario 2.1. Usando nuovamente il Teorema 2.8 otteniamo la tesi.

□

Fissato un ordine di approssimazione μ quello che vorremmo fare è minimizzare la norma dell'errore, ovvero scegliere k e m con $k+m+1 = \mu$ che realizzino il minimo della funzione $\|E((1-X)^p,k,m,X)\|$.

Teorema 2.10. Siano k e $m \geq 1$ con $k-m+1 \geq 0$, $0 \leq \alpha \leq \gamma$ e $\|\cdot\|$ una norma indotta tale che se $M_1 \leq M_2$ componente per componente allora $\|M_1\| \leq \|M_2\|$. Se $X \in \mathbb{R}^{n\times n}$ con $\|X\| \leq 1$, allora

$$\big\|E({}_2F_1(\alpha,1;\gamma;X),k,m,X)\big\| \leq \big\|E({}_2F_1(\alpha,1;\gamma;X),k+1,m-1,X)\big\|$$

Dimostrazione. Usando il Teorema 2.8 e alcune manipolazioni algebriche abbiamo

$$\begin{aligned} E({}_2F_1(\alpha,1;\gamma;x),k,m,X) &= \frac{Q_{km}(I)}{Q_{km}(X)} \sum_{j=k+m+1}^{+\infty} \frac{(\alpha)_j (j-(k+m))_m}{(\gamma)_j (j+\alpha+m)_m} X^j = \\ &= \frac{Q_{km}(I)}{Q_{km}(X)} \frac{Q_{k+1m-1}(I)}{Q_{k+1m-1}(I)} \frac{Q_{k+1m-1}(X)}{Q_{k+1m-1}(X)} \sum_{j=k+m+1}^{+\infty} \frac{(\alpha)_j (j-(k+m))_{m-1}}{(\gamma)_j (j+\alpha+m)_{m-1}} \frac{(j-k-1)}{(j+\alpha-m)} X^j \end{aligned} \tag{2.30}$$

Per il Teorema 2.7, detti $x_1,\dots,x_m$ e $z_1,\dots,z_{m-1}$ gli zeri dei polinomi Q_{km} e Q_{k+1m-1} rispettivamente, abbiamo

$$\begin{aligned} Q_{km}(x)^{-1} Q_{k+1m-1}(x) &= \frac{\prod_{j=1}^{m} x_j}{\prod_{j=1}^{m}(x_j-x)} \frac{\prod_{j=1}^{m-1}(z_j-x)}{\prod_{j=1}^{m-1} z_j} = \\ &= \frac{x_m}{x_m-x} \sum_{j=1}^{m-1} \frac{x_j(z_j-z)}{z_j(x_j-x)} = \frac{x_m}{x_m-x} \sum_{j=1}^{m-1} \frac{x_j}{z_j} \frac{(z_j-x)}{(x_j-x)} \end{aligned} \tag{2.31}$$

Poiché abbiamo visto che $1 < x_1 < z_1 < x_2 < \cdots < x_{m-1} < z_{m-1} < x_m$, da (2.31) abbiamo che $Q_{km}(x)^{-1}Q_{k+1m-1}(x)$ è positiva e crescente per x in $(0,1)$ e quindi

$$\begin{aligned} 0 &\leq Q_{km}(x)^{-1} Q_{k+1m-1}(x) Q_{km}(1) Q_{k+1m-1}(1)^{-1} \\ &\leq Q_{km}(1)^{-1} Q_{k+1m-1}(1) Q_{km}(1) Q_{k+1m-1}(1)^{-1} = 1 \end{aligned} \tag{2.32}$$

A questo punto, poiché $\|X\| \leq 1$ abbiamo $\|(z_jI-X)(x_jI-X)^{-1}\| \leq \dfrac{z_j-\|X\|}{x_j-\|X\|}$ e possiamo quindi applicare (2.32) anche nel caso matriciale. Riprendendo (2.30), passando alle norme e sostituendo (2.32) otteniamo

$$\begin{aligned} &\|E({}_2F_1(\alpha,1;\gamma;x),k,m,X)\| \leq \\ &\leq \left\| Q_{k+1m-1}(X)^{-1} Q_{k+1m-1}(I) \sum_{j=k+m+1}^{+\infty} \frac{(\alpha)_j (j-(k+m))_{m-1}}{(\gamma)_j (j+\alpha+m)_{m-1}} \frac{(j-k-1)}{(j+\alpha-m)} X^j \right\| = \\ &= \|E({}_2F_1(\alpha,1;\gamma;x),k+1,m-1,X)\| \end{aligned}$$

□

Applicando il teorema appena dimostrato nel caso $\alpha = -p$ per $p \in (0,1)$ e $\gamma = 1$ otteniamo

$$\big\|E\big((I-X)^p,k,m,X\big)\big\| \leq \big\|E\big((I-X)^p,k+1,m-1,X\big)\big\|$$

da cui possiamo concludere che l'errore di approssimazione, in norma, si riduce quando k e m sono vicini. D'ora in avanti useremo sempre l'approssimante di Padé diagonale $\mathcal{R}_m := \mathcal{R}_{mm}$.
Vogliamo ora scrivere una rappresentazione dell'approssimante $[m/m]$ di $(1-x)^p$. L'idea è quella di sfruttare la corrispondenza con le funzioni ipergeometriche per trovare una scrittura in frazione continua, poi troncare tale frazione all'$m-$esimo convergente, ottenendo così l'approssimante cercata.
Utilizzeremo le regole di adiacenza per le funzioni ipergeometriche:

$$ {}_2F_1(\alpha, \beta+1; \gamma+1; x) - {}_2F_1(\alpha, \beta; \gamma; x) = \frac{\alpha(\gamma-\beta)}{\gamma(\gamma+1)} {}_2F_1(\alpha+1, \beta+1; \gamma+2; x) \tag{2.33} $$

$$ {}_2F_1(\alpha+1, \beta; \gamma+1; x) - {}_2F_1(\alpha, \beta; \gamma; x) = \frac{\beta(\gamma-\alpha)}{\gamma(\gamma+1)} {}_2F_1(\alpha+1, \beta+1; \gamma+2; x) \tag{2.34} $$

Da (2.33) si ottiene

$$ \frac{{}_2F_1(\alpha, \beta+1; \gamma+1; x)}{{}_2F_1(\alpha, \beta; \gamma; x)} = \frac{1}{1 - x\dfrac{\alpha(\gamma-\beta)}{\gamma(\gamma+1)} \dfrac{{}_2F_1(\alpha+1, \beta+1; \gamma+2; x)}{{}_2F_1(\alpha, \beta+1; \gamma+1; x)}} = $$

utilizzando in modo analogo (2.34) per sostituire il rapporto di ipergeometriche a destra dell'uguaglianza otteniamo

$$ = \cfrac{1}{1 - x\dfrac{\alpha(\gamma-\beta)}{\gamma(\gamma+1)} \cfrac{1}{1 - x\dfrac{(\beta+1)(\gamma+1-\alpha)}{(\gamma+1)(\gamma+2)} \dfrac{{}_2F_1(\alpha+1, \beta+2; \gamma+3; x)}{{}_2F_1(\alpha+1, \beta+1; \gamma+2; x)}}} $$

Procedendo ricorsivamente, sostituendo ad ogni passaggio il rapporto di funzioni ipergeometriche usando alternativamente (2.33) e (2.34) otteniamo

$$ \frac{{}_2F_1(\alpha, \beta+1; \gamma+1; x)}{{}_2F_1(\alpha, \beta; \gamma; x)} = \cfrac{1}{1 + \cfrac{a_1 x}{1 + \cfrac{a_2 x}{\cdots}}} \tag{2.35} $$

dove

$$ a_{2n} = -\frac{(\beta+n)(\gamma-\alpha+n)}{(\gamma+2n-1)(\gamma+2n)} \qquad a_{2n+1} = -\frac{(\alpha+n)(\gamma-\beta+n)}{(\gamma+2n)(\gamma+2n+1)} $$

Utilizzando (2.35) nel caso $\alpha = p$, $\beta = \gamma = 0$ e ricordando che $(1-x)^p = {}_2F_1(-p, 1; 1; x) = {}_2F_1(p, 1; 1; x)^{-1}$ abbiamo

$$ (1-x)^p = \left(\frac{{}_2F_1(p, 1; 1; x)}{{}_2F_1(p, 0; 0; x)} \right)^{-1} = 1 + \cfrac{a_1 x}{1 + \cfrac{a_2 x}{\cdots}} $$

con

$$a_{2n} = \frac{p-n}{2(2n-1)} \qquad a_{2n+1} = \frac{p+n}{2(2n+1)}$$

e l'approssimante di Padé $[m/m]$ sarà quindi data da

$$\mathcal{R}_m(x) = 1 + \cfrac{a_1 x}{1 + \cfrac{a_2 x}{1 + \cfrac{\cdots}{1 + a_{2m} x}}} \tag{2.36}$$

Questa scrittura ci permette di valutare $\mathcal{R}_m$ in $X \in \mathbb{C}^{n\times n}$ in modo efficiente, seguendo un procedimento dal basso verso l'alto. L'algoritmo che implementa questo metodo è il seguente.

Algoritmo 4: data $X \in \mathbb{C}^{n\times n}$ valuta $\mathcal{R}_m$ dato da (2.36) dal basso verso l'alto

1: Y_{2m}=$a_{2m}X$
2: **for** $j = 2m-1 : 1$ **do**
3: ricava Y_j da $(I + Y_{j+1})Y_j = a_j X$
4: **end for**
5: $\mathcal{R}_m = I + Y_1$

Analizziamo brevemente la stabilità numerica dell'algoritmo appena presentato.
Supponiamo $\|Y_j\| \leq 1$ per ogni j, chiamiamo $\tilde{Y}_j = Y_j + \Delta Y_j$ la $j-$esima matrice effettivamente calcolata, e supponiamo che il risolutore di sistemi lineari sottostante sia stabile all'indietro per sistemi con un singolo termine noto. Per la nostra successione di sistemi avremo quindi

$$(I - \tilde{Y}_{j+1})\tilde{Y}_j = a_j X + F_j + R_j \tag{2.37}$$

con $\|F_j\| \leq u|a_j|\|X\|$ e $\|R_j\| \leq \alpha u(1+\|\tilde{Y}_j+1\|)\|\tilde{Y}_j\|$ per α costante e u unità di macchina. Una limitazione superiore alla norma di Y_j è data da

$$\begin{cases} \|Y_j\| \leq \dfrac{|a_j|\|X\|}{1 - \|Y_{j+1}\|} & 0 \leq j \leq 2m-1 \\ \|Y_{2m}\| \leq |a_{2m}|\|X\| \end{cases} \tag{2.38}$$

mentre da (2.37) otteniamo

$$(I + Y_{j+1})\Delta Y_j = F_j + R_j - \Delta Y_{j+1} Y_j - \Delta Y_{j+1}\Delta Y_j$$

e passando alle norme

$$\begin{cases} \|\Delta Y_j\| \leq \dfrac{u a_j \|X\| + \alpha u (1 + \|Y_{j+1}\|)\|Y_j\| + \|Y_j\|\|\Delta Y_{j+1}\|}{1 - \|Y_{j+1}\|} + O(u^2) \quad 0 \leq j \leq 2m-1 \\ \|\Delta Y_{2m}\| \leq u |a_{2m}| \|X\| \end{cases} \tag{2.39}$$

Le maggiorazioni (2.39) e (2.38) ci permettono di calcolare ricorsivamente una limitazione a $\|\Delta Y_1\|$, dopodiché usando il fatto che $\|Y_1\| \geq \dfrac{|a_1|\|X\|}{1 + \|Y_2\|}$ otteniamo una maggiorazione all'errore relativo $\left\| \dfrac{\Delta Y_1}{Y_1} \right\| \leq du + O(u^2)$.
Nella tabella 2.1 riportiamo i corrispondenti d per alcuni valori di $p \in (0,1)$ e $\|X\| \in (0,1)$ dove l'approssimante di Padé utilizzata è quella diagonale di indice m con $m = \min\{100, \bar{m}\}$ dove $\bar{m}$ è il minimo intero positivo per cui vale

$$|(1 - \|X\|)^p - \mathcal{R}_m(\|X\|)| \leq u \tag{2.40}$$

I corrispondenti valori di m sono riportati nella tabella 2.2. I risultati ci mostrano che finché abbiamo $\|X\| \leq 0.95$, l'algoritmo 4 è stabile.

$\|X\| \backslash p$	0.1	0.3	0.5	0.7	0.9
0.99	3.46e2	3.21e2	2.90e2	2.56e2	2.19e2
0.95	6.53e1	6.08e1	5.55e1	4.97e1	4.33e1
0.90	3.12e1	2.92e1	2.68e1	2.43e1	2.15e1
0.75	1.14e1	1.07e1	1.00e1	9.24e0	8.42e0
0.50	5.01e0	4.80e0	4.59e0	4.36e0	4.12e0
0.25	2.98e0	2.91e0	2.85e0	2.77e0	2.70e0
0.10	2.32e0	2.30e0	2.28e0	2.26e0	2.23e0

Figura 2.1: Valori di d per cui $\|\Delta Y_1/Y_1\| \leq du + O(u^2)$, corrispondenti a diversi valori di $\|X\|$ e p

$\|X\| \backslash p$	0.1	0.3	0.5	0.7	0.9
0.99	88	100	100	84	79
0.95	38	39	39	39	36
0.90	27	27	27	27	26
0.75	16	16	16	16	15
0.50	9	10	10	10	10
0.25	6	6	7	7	6
0.10	5	5	5	5	5

Figura 2.2: Minimo valore di m per cui vale la disuguaglianza (2.40)

Algoritmo

Presentiamo adesso l'algoritmo generale del metodo di Schur-Padè. Senza perdita di generalità possiamo supporre $p \in (0,1)$: per p generico basterà infatti decomporlo come $p_1 + p_2$ con $p_1 \in (0,1)$ e p_2 intero e usare la relazione $A^p = A^{p_1} A^{p_2}$. Data $A \in \mathbb{C}^{n \times n}$ non singolare senza autovalori reali negativi,chiamiamo $T = Q^H A Q$ la sua forma normale di Schur, che sarà quindi una matrice triangolare superiore, ed estraiamo ripetutamente radici quadrate di T fino ad ottenere $T^{1/2^k}$ vicina alla matrice identità. A questo punto $X := I - T^{1/2^k}$ è vicina a zero, possiamo quindi approssimare $(I-X)^p$ con l'approssimante di Padé $\mathcal{R}_m(X)$ e ottenere così $T^{p/2^k} = (I - X)^p \simeq \mathcal{R}_m(X)$ e quindi $A^p = Q T^p Q^H \simeq$

$Q\mathcal{R}_m(X)^{2^k}Q^H$.
Prima di procedere è necessario precisare cosa intendiamo quando diciamo $T^{1/2^k}$ *vicina* alla matrice identità, dobbiamo quindi stabilire una condizione che determini quante radici quadrate della matrice di partenza estrarre, tenendo conto sia della precisione di approssimazione, che delle operazioni necessarie ad estrarre ognuna delle radici quadrate. Fissati m intero e $p \in (-1, 1)$ definiamo $\theta_m^{(p)}$ come il massimo valore di $\|X\|$ per cui vale la disuguaglianza (2.40) e ricordiamo che per il Teorema 2.9 vale

$$\|(1-X)^p - \mathcal{R}_m(X)\| \leq |(1-\|X\|)^p - \mathcal{R}_m(\|X\|)| \leq u \tag{2.41}$$

Poiché si può dimostrare che per m fissato, se la distanza di p da 1, 0 e -1 è maggiore di 10^{-4}, la variazione di $\theta_m^{(p)}$ al variare di p è piccola, non è necessario ottimizzare l'algortimo per ogni specifico valore di p ma basta prendere come valore di riferimento $\theta_m := \min_{p\in[-1,1]} \theta_m^{(p)}$. Riportiamo nella tabella 2.3 alcuni valori di θ_m per $m \in \{1, \ldots, 64\}$.

m	1	2	3	4	5	6	7	8	9
θ_m	1.51e-5	2.24e-3	1.88e-2	6.04e-2	1.24e-1	2.00e-1	2.79e-1	3.55e-1	4.25e-1
m	10	11	12	13	14	15	16	32	64
θ_m	4.87e-1	5.42e-1	5.90e-1	6.32e-1	6.69e-1	7.00e-1	7.28e-1	9.15e-1	9.76e-1

Figura 2.3: Valori di $\theta_m := \min_p \theta_m^{(p)}$ per $m = 1 : 64$

Per quel che riguarda il costo, abbiamo che calcolare la radice quadrata di una matrice triangolare superiore, con l'algoritmo di Björck e Hammarling[3], richiede $\frac{1}{3}n^3$ operazioni mentre valutare $\mathcal{R}_m$ con l'algoritmo 4 richiede $\frac{1}{3}(2m-1)n^3$, quindi è vantaggioso calcolare un'ulteriore radice quadrata se permette di ridurre di più di 1 il grado dell'approssimante di Padé. Inoltre, considerando che per $T \simeq I$

$$\|I - T^{1/2}\| = \|(I + T^{1/2})^{-1}(I - T)\| \simeq \frac{1}{2}\|I - T\|$$

e che per $m > 7$ $\frac{1}{2}\theta_m < \theta_{m-2}$, il costo di calcolare T^p quando $\|I - T\| > \theta_7$ sarà minimizzato se estraiamo successivamente radici quadrate di T fino ad avere $\|I - T^{1/2^k}\| \leq \theta_7$ e procediamo ulteriormente solo se questo ci permette di ridurre il grado dell'approssimante di Padé.
L'ultimo punto saliente dell'algoritmo riguarda l'implementazione della fase in cui calcoliamo $\mathcal{R}_m(I - T^{1/2^k})^{2^j} \simeq T^{p/2^{k-j}}$ per $j = 1 : k$. L'idea è quella di calcolare direttamente gli elementi diagonali e sopradiagonali e inserirli nella ricorsione, per limitare la propagazione degli errori. Vediamo quindi come derivare una formula per gli elementi della prima sopradiagonale.

Teorema 2.11. Data $T \in \mathbb{C}^{n\times n}$ matrice triangolare superiore con elementi T_{ij} e f funzione analitica. Detta $F := f(T)$, abbiamo

$$F_{ij} = \begin{cases} 0 & i > j \\ f(T_{ii}) & i = j \\ T_{ij} f'(T_{ii}) & i < j, T_{ii} = T_{jj} \\ T_{ij} \dfrac{F_{ii} - F_{jj}}{T_{ii} - T_{jj}} + \displaystyle\sum_{i+1}^{j-1} \dfrac{F_{ik}T_{kj} - T_{ik}F_{kj}}{T_{ii} - T_{jj}} & i < j, T_{ii} \neq T_{jj} \end{cases} \tag{2.42}$$

Dimostrazione.
Dalla definizione 1.5 abbiamo

$$F_{ij} = \frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z)(zI - T)^{-1}_{ij} dz$$

Poiché T è triangolare superiore abbiamo $(zI-T)^{-1}_{ij} = 0$ per ogni $i > j$ e quindi in questo caso $F_{ij} = 0$ poiché la funzione integranda è nulla. Per quanto riguarda gli elementi diagonali, si verifica direttamente che $(zI - T)^{-1}_{ii} = \frac{1}{z-T_{ii}}$

$$F_{ii} = \frac{1}{2\pi \mathrm{i}} \int_\Gamma f(z) \frac{1}{z - T_{ii}} dz$$

che è proprio $f(T_{ii})$ per la formula integrale di Cauchy. Analogamente nel caso $i < j$ con $T_{ii} = T_{jj}$ abbiamo $(zI-T)^{-1}_{ii} = \frac{1}{(z-T_{ii})^2}$ e dalla formula integrale di Cauchy per le derivate abbiamo la tesi.
Dimostriamo il caso $i < j$ con $T_{ii} \neq T_{jj}$. Se $T_{ii} \neq T_{jj}$, poiché $F = f(T)$ è un polinomio in T abbiamo $FT = TF$ e quindi, componente per componente

$$F_{ij}T_{jj} + F_{ii}T_{ij} + \sum_{k=i+1}^{j-1} F_{ik}T_{kj} = (FT)_{ij} = (TF)_{ij} = T_{ij}F_{jj} + T_{ii}F_{ij} + \sum_{k=i+1}^{j-1} T_{ik}F_{kj}$$

da cui abbiamo

$$F_{ij}(T_{jj} - T_{ii}) = -F_{ii}T_{ij} - \sum_{k=i+1}^{j-1} F_{ik}T_{kj} + T_{ij}F_{jj} + \sum_{k=i+1}^{j-1} T_{ik}F_{kj} =$$
$$= T_{ij}F_{jj} - F_{ii}T_{ij} + \sum_{k=i+1}^{j-1} (T_{ik}F_{kj} + F_{ik}T_{kj})$$

e quindi la tesi:

$$F_{ij} = T_{ij} \frac{F_{jj} - F_{ii}}{T_{jj} - T_{ii}} + \sum_{k=i+1}^{j-1} \frac{T_{ik}F_{kj} + F_{ik}T_{kj}}{T_{jj} - T_{ii}}$$

□

Usando il teorema che abbiamo appena dimostrato nel caso $f(z) = z^p$ e chiamando λ_i gli elementi diagonali, che sappiamo essere anche gli autovalori della nostra matrice di partenza, abbiamo

$$T_{ij}^p = \begin{cases} pT_{ij}\lambda_{ij}^{p-1} & \lambda_i = \lambda_j \\ T_{ij}\dfrac{\lambda_j^p - \lambda_i^p}{\lambda_j - \lambda_i} & \lambda_i \neq \lambda_j \end{cases}$$

Nel caso $\lambda_i \neq \lambda_j$ potremmo avere problemi di cancellazione, vediamo quindi come valutare la formula quando λ_i e λ_j sono vicini.

$$\begin{aligned} &\frac{\lambda_j^p - \lambda_i^p}{\lambda_j - \lambda_i} = \frac{e^{p\ln\lambda_j} - e^{p\ln\lambda_i}}{\lambda_j - \lambda_i} = \\ &= \exp\left(\frac{p}{2}(\ln\lambda_j + \ln\lambda_i)\right)\frac{\exp(\frac{p}{2}(\ln\lambda_j - \ln\lambda_i)) - \exp(\frac{p}{2}(\ln\lambda_i - \ln\lambda_j))}{\lambda_j - \lambda_i} = \\ &= \exp\left(\frac{p}{2}(\ln\lambda_j + \ln\lambda_i)\right)\frac{\sinh\left(\frac{p}{2}(\ln\lambda_j - \ln\lambda_i)\right)}{\lambda_j - \lambda_i} \end{aligned} \tag{2.43}$$

ci siamo quindi ridotti al problema di calcolare $\ln\lambda_j - \ln\lambda_i$ con precisione.
Detta $\mathcal{U}(z) := \dfrac{z - \ln(e^z)}{2\pi\mathrm{i}}$ e ponendo $t = \dfrac{\lambda_j - \lambda_i}{\lambda_j + \lambda_i}$ abbiamo

$$\begin{aligned} &\ln\lambda_j - \ln\lambda_i = \ln\left(\frac{\lambda_j}{\lambda_i}\right) + 2\pi\mathrm{i}\mathcal{U}(\ln\lambda_j - \ln\lambda_i) = \ln\left(\frac{1+t}{1-t}\right) + 2\pi\mathrm{i}\mathcal{U}(\ln\lambda_j - \ln\lambda_i) = \\ &= 2\mathrm{arctgh}(t) + 2\pi\mathrm{i}\mathcal{U}(\ln\lambda_j - \ln\lambda_i) \end{aligned}$$

e sostituendo in (2.43) otteniamo

$$T_{ij}\frac{\lambda_2^p - \lambda_1^p}{\lambda_j - \lambda_i} = T_{ij}\exp\left(\frac{p}{2}(\ln\lambda_j + \ln\lambda_i)\right)\frac{\sinh\left(p(\mathrm{arctgh}(t) + \pi\mathrm{i}\mathcal{U}(\ln\lambda_j - \ln\lambda_i)\right)}{\lambda_j - \lambda_i} \tag{2.44}$$

La formula per gli elementi sopradiagonali può quindi essere riscritta come

$$T_{ij}^p = \begin{cases} pT_{ij}\lambda_{ij}^{p-1} & \lambda_i = \lambda_j \\ T_{ij}\dfrac{\lambda_j^p - \lambda_i^p}{\lambda_j - \lambda_i} & |\lambda_i| < \frac{1}{2}|\lambda_j| \text{ o } |\lambda_j| < \frac{1}{2}|\lambda_i| \\ (2.44) & \textit{altrimenti} \end{cases} \tag{2.45}$$

dove usiamo la formula ricavata dal Teorema 2.11 quando λ_i e λ_j sono abbastanza lontani, e (2.44) quando sono vicini. Vediamo adesso l'algoritmo completo.

Algoritmo 5: dati $A \in \mathbb{C}^{n\times n}$ senza autovalori su $\mathbb{R}^-$ e $p \in (-1, 1)$ calcola A^p attraverso l'approssimazione di Padé

1: calcola $T = Q^H AQ$ forma di Schur complessa
2: **if** (T diagonale) **then**

3: return $X = QT^pQ^H$
4: **end if**
5: $T_0 = T$, $k = 0$, $q = 0$
6: **while** (true) **do**
7: $\tau = \|I - T\|_1$
8: **if** $(\tau \leq \theta_7)$ **then**
9: $q++$
10: $j_1 = \min\{i \in \mathbb{N} | 3 \leq i \leq 7, \quad \tau \leq \theta_i\}$
11: $j_2 = \min\{i \in \mathbb{N} | 3 \leq i \leq 7, \quad \frac{1}{2}\tau \leq \theta_i\}$
12: **if** $(j_1 - j_2 \leq 1 \quad \vee \quad (q = 2 \ \wedge m = j_1))$ **then**
13: goto line 18
14: **end if**
15: **end if**
16: $T = T^{1/2}$, $k++$
17: **end while**
18: valuta $U = \mathcal{R}_m(I - T)$ usando l'algoritmo 4
19: **for** i=k:0 **do**
20: **if** $(i < k)$ **then**
21: $U = U^2$
22: **end if**
23: sostituisci diag(U) con diag$(T_0)^{p/2^i}$
24: sostituisci la sopradiagonale di U con la sopradiag. di $(T_0)^{p/2^i}$ calcolata con (2.45)
25: **end for**
26: return $X = QUQ^H$

Analisi del costo

Il calcolo della decomposizione di Schur costa $25n^3$ operazioni, la valutazione dell'approssimante con l'algoritmo 4 ne richiede $(2m-1)\frac{n^3}{3}$ mentre il processo di estrazione delle radici e di elevamento al quadrato dell'approssimazione circa $2k\frac{n^3}{3}$. Considerando che la composizione di X richiederà $3n^3$ operazioni otteniamo che il costo totale dell'algoritmo è di $(28 + \frac{1}{3}(2k + 2m - 1))n^3$.

Generalizzazioni

Dato un generico $p \in \mathbb{R}$ possiamo applicare l'algoritmo descritto scomponendo p come somma di un intero e di una parte frazionaria ottenendo $p = \lfloor p \rfloor + p_1$ con $p_1 \in (0, 1)$ oppure $p = \lceil p \rceil + p_2$ con $p_2 \in (-1, 0)$. A questo punto basterà applicare l'algoritmo di Schur Padé per calcolare A^{p_1} oppure A^{p_2} e usare il fatto che $A^p = A^{\lfloor p \rfloor}A^{p_1} = A^{\lceil p \rceil}A^{p_2}$. La scelta tra le due decomposizioni è basata sul condizionamento del calcolo di A^{p_1} e A^{p_2}: si

dimostra che per A Hermitiana e definita positiva con autovalori $\lambda_1 \geq \cdots \geq \lambda_n > 0$

$$\|L_{x^p}(A)\|_F = |p|\lambda_n^{p-1} \tag{2.46}$$

e quindi da (2.24), ricordando che $\mathcal{C}_2(A) = \|A\|_2\|A^{-1}\|_2$ abbiamo

$$\mathcal{C}_{x^p}(A) = \frac{|p|\lambda_n^{p-1}\|A\|_F}{\|A^p\|_F} \simeq \frac{|p|\lambda_n^{p-1}\|A\|_2}{\|A^p\|_2} = \begin{cases} |p|\mathcal{C}_2(A)^{1-p} & p \geq 0 \\ |p|\mathcal{C}_2(A) & p \leq 0 \end{cases} \tag{2.47}$$

sceglieremo quindi la prima decomposizione quando $p_1\mathcal{C}_2(A)^{1-p_1} \leq -p_2\mathcal{C}_2(A)$ e la seconda altrimenti.

2.2.3 Algoritmo di Schur-Padé migliorato

In questo paragrafo, facendo riferimento all'articolo di Higham e Lin [7], presenteremo un'estensione del metodo di Schur-Padé con tre scopi principali: migliorare l'accuratezza e l'efficienza affinando l'analisi dell'errore, adattare l'algoritmo al calcolo delle derivate di Fréchet e modificarlo in modo che per matrici reali utilizzi solo aritmetica reale.

Analisi dell'errore

Abbiamo dimostrato che data $X \in \mathbb{C}^{n\times n}$ con $\|X\| \leq 1$, $p \in (-1, 1)$ e $k \geq m$ l'errore di approssimazione è limitato superiormente da

$$\|(I - X)^p - \mathcal{R}_{km}(X)\| \leq |(1 - \|X\|)^p - \mathcal{R}_{km}(\|X\|)| \tag{2.48}$$

mostreremo che è possibile migliorare tale maggiorazione. Per $p \in \mathbb{N}^+$ definiamo

$$\alpha_p(X) := \max\{\|X^p\|^{1/p}, \|X^{p+1}\|^{1/p+1}\}$$

e osserviamo che, detto $\rho(X)$ il raggio spettrale della matrice, usando la submoltiplicatività della norma, si ha

$$\rho(X) \leq \alpha_p(X) \leq \|X\|.$$

Teorema 2.12. Siano $h_l(x) := \sum_{j=l}^{+\infty} c_j x^j$ con raggio di convergenza ω e $\tilde{h}_l(x) := \sum_{j=l}^{+\infty} |c_j| x^j$. Se $X \in \mathbb{C}^{n\times n}$ con $\rho(X) \leq \omega$ allora $\|h_l(X)\| \leq \tilde{h}_l(\alpha_p(X))$ per ogni p tale che $p(p-1) \leq l$

Dimostrazione.
Per iniziare consideriamo $k = pm_1 + qm_2 \quad \text{con} \quad m_1, m_2 \in \mathbb{N}$

$$\|X^k\| = \|X^{pm_1+qm_2}\| \leq \|X^p\|^{m_1}\|X^q\|^{m_2} = (\|X^p\|^{1/p})^{pm_1}(\|X^q\|^{1/q})^{qm_2} \leq$$
$$\leq \max\left(\{\|X^p\|^{1/p}, \|X^q\|^{1/q}\}\right)^{pm_1+qm_2}$$

ovvero

$$\|X^k\|^{1/k} \leq \max\{\|X^p\|^{1/p}, \|X^q\|^{1/q}\} \tag{2.49}$$

Definiamo

$$J_p := \{m \in \mathbb{N} | m \geq p(p-1)\} \quad \text{e} \quad Y_p := \{(p+1)m_1 + pm_2 | m_1, m_2 \in \mathbb{N}\}$$

si verifica facilmente che $J_p \subseteq Y_p$ e quindi per (2.49) se $k \in J_p$ abbiamo

$$\|X^k\|^{1/k} \leq \max\{\|X^p\|^{1/p}, \|X^{p+1}\|^{1/p+1}\} = \alpha_p(X). \tag{2.50}$$

Possiamo adesso dimostrare la disuguaglianza cercata:

$$\|h_l(X)\| \leq \sum_{j=l}^{+\infty} |c_j| \|X^j\| = \sum_{j=l}^{+\infty} |c_j| (\|X^j\|^{1/j})^j \leq \sum_{j=l}^{+\infty} |c_j| (\max\{\|X^k\|^{1/k} | k \geq l\})^j \tag{2.51}$$

ma $l \geq p(p-1)$ per ipotesi, quindi ognuno degli interi k su cui calcoliamo il massimo nell'ultima sommatoria appartiene a J_p e usando (2.50) otteniamo

$$\left(\max\{\|X^k\|^{1/k} | k \geq l\}\right)^j \leq \alpha_p(X)^j$$

e quindi riprendendo la disuguaglianza (2.51) abbiamo

$$\|h_l(X)\| \leq \sum_{j=l}^{+\infty} |c_j| \alpha_p(X)^j = \tilde{h}_l(\alpha_p(X))$$

□

Vediamo ora un risultato che ci permette di dare una maggiorazione dell'errore di approssimazione in termini di α_p

Teorema 2.13. Sia $X \in \mathbb{C}^{n \times n}$ con $\rho(X) < 1$,per ogni $t \in (-1, 1)$ e $k, m \in \mathbb{N}$ con $k \geq m$ abbiamo

1. $(I - X)^t - \mathcal{R}_{km}(X) = \sum_{j=k+m+1}^{+\infty} \psi_j X^j$ dove i coefficienti ψ_j hanno tutti lo stesso segno
2. $\|(I - X)^t - \mathcal{R}_{km}(X)\| \leq |(1 - \alpha_p(X))^t - \mathcal{R}_{km}(\alpha_p(X))|$ per $p(p-1) \leq k + m + 1$

Dimostrazione.
Utilizzando il Teorema 2.8 e alcune manipolazioni algebriche otteniamo la prima parte della tesi. Per quanto riguarda la seconda parte, utilizzando 1. e il teorema precedente abbiamo

$$\begin{aligned} \|(I - X)^t - \mathcal{R}_{km}(X)\| &= \left\| \sum_{j=k+m+1}^{+\infty} \psi_j X^j \right\| \leq \sum_{j=k+m+1}^{+\infty} |\psi_j| \alpha_p(X)^j = \\ &= \left| \sum_{j=k+m+1}^{+\infty} \psi_j \alpha_p(X)^j \right| = |(1 - \alpha_p(X))^t - \mathcal{R}_{km}(\alpha_p(X))| \end{aligned} \tag{2.52}$$

che è proprio la disuguaglianza 2.

□

La prima differenza sostanziale tra l'algoritmo di Schur-Padé presentato nel paragrafo precedente e questa nuova versione, sta nella disuguaglianza che prendiamo come riferimento per assicurarci la precisione cercata nell'approssimazione di Padé: l'algoritmo di Schur-Padé si basava sulla norma della matrice $I - T^{1/2^k}$ sfruttando la maggiorazione (2.48), questa versione migliorata utilizza il limite superiore dell'errore dato da

$$\|(I - X)^t - \mathcal{R}_{km}(X)\| \leq |(1 - \alpha_p(X))^t - \mathcal{R}_{km}(\alpha_p(X))| \tag{2.53}$$

e considera come riferimento la quantità $\alpha_p(I - X^{1/2^k})$ che in generale è più piccola di $\|I - T^{1/2^k}\|$. Vediamo adesso come scegliere il numero s di radici estratte dalla matrice di partenza e l'indice dell'approssimante di Padé m, in modo da minimizzare il costo dell'algoritmo rispettando la condizione di accuratezza richiesta $|(1 - \alpha_p(X))^t - \mathcal{R}_{km}(\alpha_p(X))| \leq u$, dove u è l'unità di macchina. Data una coppia di valori s e m, che verificano $\alpha_p(I - T^{1/2^s}) \leq \theta_m$, analogamente a quanto visto nel caso precedente, è conveniente calcolare un'ulteriore radice quadrata di T se permette di abbassare l'indice m di più di 1, ovvero se $\alpha_p(I - T^{1/2^{s+1}}) \leq \theta_{m-2}$ per qualche p, inoltre, poiché per s abbastanza grande $\alpha_p(I - T^{1/2^{s+1}}) \simeq \frac{1}{2}\alpha_p(I - T^{1/2^s})$ e, come abbiamo osservato nel paragrafo precedente $\frac{1}{2}\theta_m < \theta_{m-2}$ per $m > 7$, il costo sarà minimizzato se estraiamo radici quadrate di T fino ad ottenere $\alpha_p(I - T^{1/2^s}) \leq \theta_7$ per qualche $p \in \{1, 2, 3, 4\}$.
Se la matrice T è triangolare superiore e $D = \mathrm{diag}(T)$, allora $\rho(I - D^{1/2^s}) = \rho(I - T^{1/2^s}) \geq \alpha_p(I - T^{1/2^s})$ e poiché calcolare $\rho(I - D^{1/2^s})$ richiede ovviamente meno operazioni di calcolare $\alpha_p(I - T^{1/2^s})$, possiamo ottenere un ulteriore miglioramento nell'efficienza, infatti ogni volta che calcoliamo una nuova radice di T possiamo calcolare $\rho(I - D^{1/2^s})$ e calcolare esplicitamente $\alpha_p(I - T^{1/2^s})$ solo quando abbiamo $\rho(I - D^{1/2^s}) \leq \theta_7$.
Ricapitoliamo il procedimento che seguiremo, a partire da T triangolare superiore. Estraiamo radici quadrate di T fino ad ottenere $T^{1/2^{\bar{s}}}$ tale che $\rho(I - D^{1/2^{\bar{s}}}) \leq \theta_7$ dopodiché calcoliamo $T^{1/2^s}$. Se $\alpha_p(I - T^{1/2^s}) \leq \theta_i$ con $i = 1, 2$ allora prendiamo come approssimante di Padé proprio $\mathcal{R}_i$ e non sono necessarie altre radici, altrimenti dobbiamo individuare quante ulteriori radici calcolare e quale approssimante di Padé utilizzare. Abbiamo già osservato che il massimo indice di approssimazione che dobbiamo considerare è $m = 7$, controlleremo quindi soltanto α_3 e α_4. Se $\alpha_3(I - T^{1/2^s}) > \theta_7$ e $\alpha_4(I - T^{1/2^s}) > \theta_7$ calcoliamo un'altra radice e ripetiamo le considerazioni appena fatte, se $\alpha_3(I - T^{1/2^s}) > \theta_7$ e $\alpha_4(I - T^{1/2^s}) \leq \theta_7$ allora, se $\alpha_3(I - T^{1/2^s}) \leq \theta_6$ poniamo $m = 6$ mentre se $\alpha_4(I - T^{1/2^s}) > \theta_6$ prenderemo $m = 7$. Resta da considerare solo il caso in cui $\alpha_3(I - T^{1/2^s}) \leq \theta_7$: se $\alpha_3(I - T^{1/2^s}) > \theta_6$ e $\frac{1}{2}\alpha_3(I - T^{1/2^s}) \leq \theta_5$ allora per quanto abbiamo osservato è conveniente calcolare un'ulteriore radice, ma poiché non c'è garanzia che avremo $\alpha_3(I - T^{1/2^s}) \leq \theta_5$, poniamo a 2 il numero massimo di ulteriori radici che possiamo calcolare; se se $\alpha_3(I - T^{1/2^s}) > \theta_6$ e $\frac{1}{2}\alpha_3(I - T^{1/2^s}) > \theta_5$ consideriamo $\alpha_4(I - T^{1/2^s})$ come sopra per distinguere tra $m = 6$ e $m = 7$; infine se $\alpha_3(I - T^{1/2^s}) \leq \theta_6$ non serve un'ulteriore radice, cerchiamo quindi il minimo $m \in \{3, 4, 5, 6\}$ tale che $\alpha_3(I - T^{1/2^s}) \leq \theta_m$.

Vediamo l'algoritmo di Schur-Padé in cui abbiamo apportato le modifiche descritte fino ad ora. Usiamo la funzione di Matlab `normest`(A,m) che fornisce una stima del valore $\|A^m\|_1$

Algoritmo 6: dati $A \in \mathbb{C}^{n\times n}$ senza autovalori su $\mathbb{R}^-$ e $p \in (-1, 1)$ calcola A^p attraverso l'approssimazione di Padé

1: calcola $T = Q^H AQ$ forma di Schur complessa
2: **if** (T diagonale) **then**
3: return $X = QT^pQ^H$
4: **end if**
5: $T_0 = T$, $D = \mathrm{diag}(T)$
6: trovo $\bar{s} := \min\{s \mid \rho(I - D^{1/2^s}) \leq \theta_7\}$
7: **for** $j = 1 : \bar{s}$ **do**
8: $T = T^{1/2}$
9: **end for**
10: $s = \bar{s}$, $q = 0$
11: $d_2 = \mathtt{normest}(I - T, 2)^{1/2}$, $d_3 = \mathtt{normest}(I - T, 3)^{1/3}$, $\alpha_2 = \max\{d_2, d_3\}$
12: **for** $i = 1 : 2$ **do**
13: **if** $(\alpha_2 \leq \theta_i)$ **then**
14: $m = i$, go to line 42
15: **end if**
16: **end for**
17: **while** (true) **do**
18: **if** $(s > \bar{s})$ **then**
19: $\mathtt{normest}(I - T, 3)^{1/3}$
20: **end if**
21: $d_4 = \mathtt{normest}(I - T, 4)^{1/4}$, $\alpha_3 = \max\{d_3, d_4\}$
22: **if** $(\alpha_3 \leq \theta_7)$ **then**
23: $j_1 = \min\{i \in \{3, \ldots, 7\} \mid \alpha_3 \leq \theta_i\}$
24: **if** $(j_1 \leq 6)$ **then**
25: $m = j_1$ go to line 42
26: **else**
27: **if** $(\frac{1}{2}\alpha_3 \leq \theta_5$ **and** $q < 2)$ **then**
28: $q++$, go to line 42
29: **end if**
30: **end if**
31: **end if**
32: $d_5 = \mathtt{normest}(I - T, 5)^{1/5}$, $\alpha_4 = \max\{d_4, d_5\}$, $\eta = \min\{\alpha_3, \alpha_4\}$
33: **for** i=6:7 **do**
34: **if** $(\eta\theta_i)$ **then**
35: $m = i$, go to line 42

36: **end if**
37: **end for**
38: $T = T^{1/2}$, $s++$
39: **end while**
40: $R = I - T$
41: sostituisci la diagonale e la prima sopradiagonale di R con quelle di $(I - T_0^{p/2^i})$ calcolate direttamente
42: valuta $U = \mathcal{R}_m(R)$ usando l'algoritmo 4
43: **for** $j = s : 0$ **do**
44: **if** $(i < s)$ **then**
45: $U = U^2$
46: **end if**
47: sostituisci la diagonale e la prima sopradiagonale di U con quelle di $T_0^{p/2^i}$ calcolate direttamente
48: **end for**
49: $X = QUQ^H$

Derivate di Fréchet

Vogliamo adesso modificare l'algoritmo in modo da calcolare la derivata di Fréchet di $f(x) = x^t$ in A. L'idea è quella di calcolare simultaneamente A^t e $L_{x^t}(A)$ riutilizzando per le derivate le quantità già calcolate per le potenze.
Differenziando $A^t = (A^2)^{t/2} = (A^{t/2})^2$ e ponendo $E_1 := L_{x^{1/2}}(A, E)$ otteniamo

$$\begin{aligned} &L^2_{(x^{t/2})}(A, E) = A^{t/2} L_{x^{t/2}}(A, E) + L_{x^{t/2}}(A, E) A^{t/2} = \\ &= A^{t/2} L_{x^t}(A^{1/2}, L_{x^{1/2}}(A, E)) + L_{x^t}(A^{1/2}, L_{x^{1/2}}(A, E)) A^{t/2} = \\ &= A^{t/2} L_{x^t}(A^{1/2}, E_1) + L_{x^t}(A^{1/2}, E_1) A^{t/2} \end{aligned} \tag{2.54}$$

e inoltre

$$A^{1/2} E_1 + E_1 A^{1/2} = L_{x^{1/2} x^{1/2}}(A, E) = L_x(A, E) = E \tag{2.55}$$

Queste uguaglianze ci permettono di calcolare $L_{x^t}(A, E)$ sfruttando alcune ricorrenze:

$$\begin{cases} X_0 = A \\ X_k = X_{k-1}^{1/2} \qquad k = 1 : s \end{cases} \qquad \begin{cases} E_0 = A \\ E_k X_k + X_k E_k = E_{k-1} \qquad k = 1 : s \end{cases}$$

al termine di questa prima coppia abbiamo $X_s = A^{1/2^s}$ e $E_s = L_{x^{1/2^s}}(A, E)$ definiamo quindi

$$\begin{cases} Y_0 = \mathcal{R}_m(I - X_s) \\ Y_{k-1} = Y_k^2 \qquad\qquad k = s - 1 : 1 \end{cases} \qquad \begin{cases} L_s \simeq L_{x^t}(X_s, E_s) \\ L_{k-1} = Y_k L_k + L_k Y_k \qquad k = s - 1 : 1 \end{cases}$$

dove, poiché

$$L_{(1-x)^t}(I-X,-E) = L_{x^t}(I-(I-X), L_{(1-x)}(I-X,E)) = L_{x^t}(X,E)$$

prendiamo come approssimante di $L_{x^t}(X_s,E_s)$ proprio $L_{R_m}(I-X_s,-E_s)$.
Studiamo l'errore di approssimazione e vediamo come valutare L_{R_m}. Dal Teorema 2.13 abbiamo che $(I-X)^t - \mathcal{R}_m(X) = \sum_{j=2m+1}^{+\infty} \psi_j X^j =: h_{2m+1}(X)$ e differenziando entrambi i membri otteniamo

$$L_{(1-x)^t}(X,E) - L_{R_m}(X,E) = \sum_{j=2m+1}^{+\infty} \psi_j L_{x^j}(X,E) = \sum_{j=2m+1}^{+\infty} \psi_j \sum_{k=1}^{j} X^{k-1} E X^{j-k}$$

dove l'ultima uguaglianza si dimostra usando ricorsivamente il fatto che $L_{x^k}(X,E) = L_{xx^{k-1}}(X,E) = XL_{x^{k-1}} + L_x(X,E)X^{k-1}$ e $L_x(X,E) = E$. Passando alle norme otteniamo una maggiorazione all'errore di approssimazione della derivata di Fréchet:

$$\begin{aligned}
&\|L_{(1-x)^t}(X,E) - L_{R_m}(X,E)\| \leq \sum_{j=2m+1}^{+\infty} |\psi_j| \sum_{k=1}^{j} \|X^{k-1} E X^{j-k}\| \leq \\
&\leq \sum_{j=2m+1}^{+\infty} |\psi_j| \sum_{k=1}^{j} \|X\|^{j-1} \|E\| \leq \left| \sum_{j=2m+1}^{+\infty} j\psi_j \|X\|^{j-1} \|E\| \right| = \left| h'_{2m+1}(\|X\|) \right| \|E\|
\end{aligned}$$

Per valutare L_{R_m} partiamo dalla rappresentazione in frazione continua di R_m vista in (2.36) e definiamo

$$\begin{cases} y_{2m}(x) = a_{2m}x \\ y_j(x) = \dfrac{a_j x}{1 + y_{j+1}(x)} \end{cases} \qquad j = 2m-1, \ldots, 1$$

osservando che la seconda equazione equivale ovviamente a $(I + y_{j+1}(X))y_j(X) = a_j X$ e differenziando entrambe le uguaglianze otteniamo

$$\begin{cases} L_{y_{2m}}(X,E) = a_{2m}E \\ L_{y_{j+1}}(X,E)y_j(X) + (I + y_{j+1}(X))L_{y_j}(X,E) = a_j E \end{cases} \qquad j = 2m-1, \ldots, 1$$

da cui possiamo ricavare $L_{y_1}(X,E) = L_{\mathcal{R}_m}(X,E)$. Vediamo un algoritmo che sfrutta quanto detto fino ad ora per valutare contemporaneamente $\mathcal{R}_m$ e $L_{\mathcal{R}_m}$.

Algoritmo 7: data $X \in \mathbb{C}^{n\times n}$ valuta $\mathcal{R}_m$ e $L_{\mathcal{R}_m}$ dal basso verso l'alto

1: $Y_{2m} = a_{2m}X$, $Z_{2m} = a_{2m}E$
2: **for** $j = 2m-1 : 1$ **do**
3: ricava Y_j da $(I + Y_{j+1})Y_j = a_j X$
4: ricava Z_j da $(I + Y_{j+1})Z_j = a_j E - Z_{j+1}Y_j$

5: **end for**
6: $\mathcal{R}_m = I + Y_1$
7: $L_{\mathcal{R}_m} = Z_1$

Utilizzando il procedimento appena presentato possiamo modificare l'algoritmo 6 ottenendo un metodo che calcoli sia A^t che la derivata $L_{x^t}(A, E)$.

Caso reale

Date $A, E \in \mathbb{R}^{n\times n}$ sappiamo che anche le matrici A^p e $L_{x^p}(A, E)$ sono reali, vogliamo quindi modificare l'algoritmo in modo che in questo caso utilizzi soltanto aritmetica reale, migliorando così l'efficienza e l'accuratezza del metodo. Per iniziare, data A la matrice in input, sostituiremo il calcolo della forma normale di Schur complessa con quella reale: l'algoritmo lavorerà quindi non più su una matrice triangolare superiore, ma su T matrice triangolare a blocchi, i cui blocchi diagonali rappresentano gli autovalori della matrice di partenza ed hanno quindi dimensione 1 nel caso di autovalori reali oppure 2 nel caso di una coppia di autovalori complessi coniugati. Come conseguenza immediata di questo cambiamento, dobbiamo rivedere la fase in cui inseriamo nel calcolo gli elementi diagonali e sopradiagonali calcolati direttamente (riga 47 Alg.6): calcoliamo e sostituiamo ogni blocco diagonale ricordando che se T_{ij} rappresenta la coppia di autovalori $\theta \pm \mu$ la sua potenza p-esima sarà proprio il blocco che rappresenta $(\theta \pm \mu)^p$, se sono presenti più blocchi di diagonali adiacenti di dimensione 1 calcoliamo l'elemento sopradiagonale corrispondente usando la formula (2.45) vista per il caso complesso.
L'ultima importante differenza riguarda l'algoritmo ausiliario usato per calcolare le radici quadrate della matrice T: nel caso di matrici reali, infatti, sostituiamo l'algoritmo di Björk e Hammarling con un algoritmo che usa esclusivamente aritmetica reale (si veda Higham [8]).

Capitolo 3

Risultati Numerici

usiamo la Presentiamo ora alcuni test effettuati sui metodi di Iannazzo-Manasse e Schur-Padé: per entrambi gli algoritmi abbiamo usato le implementazioni in Matlab fornite direttamente dagli autori. Chiamiamo A la matrice di partenza e $\tilde{Y}$ la radice p-esima effettivamente calcolata dall'algoritmo, salvo dove indicato diversamente useremo come misura dell'accuratezza dei metodi la quantità $\rho_A(\tilde{Y}) := \dfrac{\|A - \tilde{Y}^p\|}{\|\tilde{Y}\| \|\sum_{i=0}^{p-1} (\tilde{Y}^T)^{p-i-1} \otimes \tilde{Y}^i\|}$.

Test 1

Abbiamo preso una matrice reale random 50×50, quasi triangolare superiore e senza autovalori a parte reale negativa e abbiamo calcolato la radice p-esima per tutti i valori di p compresi tra 10 e 300 usando entrambi i metodi e tenendo traccia del tempo necessario per ogni p. Nelle figure 3.1 e 3.2 riportiamo i grafici raffiguranti i tempi impiegati dai due algoritmi.

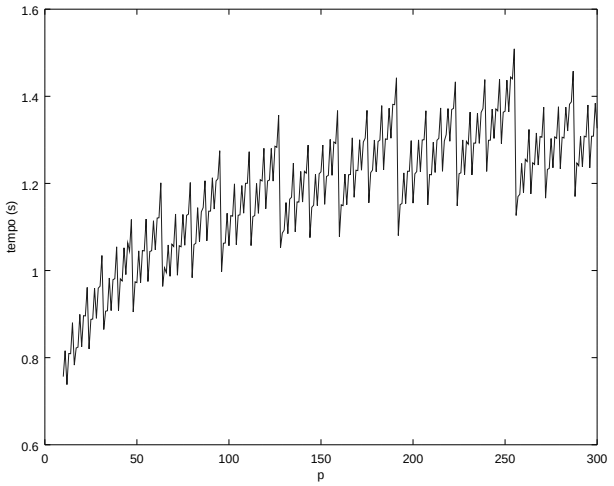

Figura 3.1: Metodo di Iannazzo-Manasse

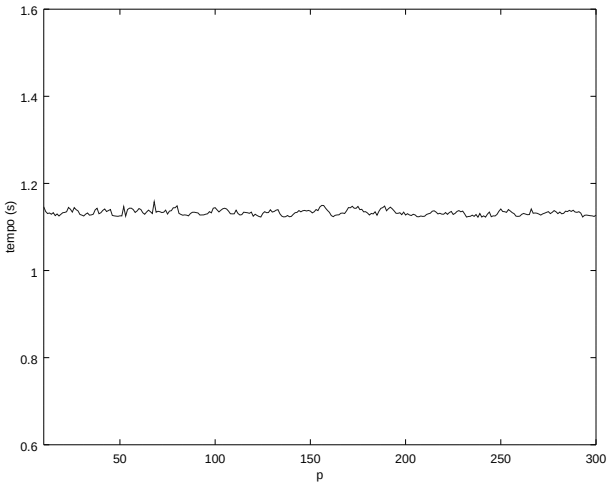

Figura 3.2: Metodo di Schur-Padé

Nel caso del metodo di Iannazo-Manasse il tempo ha crescita circa logaritmica rispetto a p, con varie oscillazioni dovute al fatto che il costo computazionale dell'algoritmo dipende dal numero di cifre uguali a 1 nell'espansione binaria di p: abbiamo quindi prestazioni ottime nel caso in cui p sia una potenza di 2 e pessime nel caso in cui p sia dato da $2^k - 1$. Nella tabella 3.3 riportiamo il tempo impiegato dal metodo di Iannazzo-Manasse per alcune coppie di valori consecutivi di p della forma $(2^k - 1, 2^k)$. Il tempo impiegato dal metodo di Schur-Padé risulta sostanzialmente costante al variare di p.

p	15	16	31	32	63	64	127	128	255	256
t	0.88019	0.78294	1.03457	0.86466	1.20136	0.96361	1.35678	1.05193	1.50899	1.12641

Figura 3.3: Confronto per valori di p consecutivi

Test 2

Il seguente test mostra la dipendenza del tempo di esecuzione dei due algoritmi dalla dimensione della matrice. Abbiamo preso una matrice $n \times n$ della forma

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}$$

e abbiamo calcolato $A^{1/10}$ per alcuni valori di n tra 10 e 500. Nel grafico 3.4 riportiamo i tempi impiegati dai due algoritmi in secondi, in funzione della dimensione n. Notiamo che mentre per valori piccoli della dimensione, fino a $n = 120$, il tempo di esecuzione dei due algoritmi è circa lo stesso, per matrici di dimensione superiore il metodo di Schur-Padé risulta migliore, in termini di tempo impiegato, di quello di Iannazzo-Manasse.

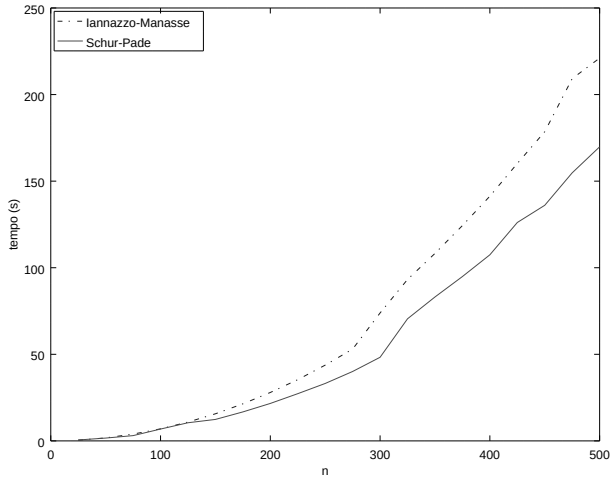

Figura 3.4: Indichiamo con la linea tratteggiata l'algoritmo di Iannazzo-Manasse e con la linea continua l'algoritmo di Schur-Padé

Test 3

Data la matrice A= $\begin{pmatrix} 1 & 1 \\ 0 & 1+10^{-t} \end{pmatrix}$ calcoliamo $A^{q/p}$ per $(p,q) \in \{(10,1),(2,1),(9,10)\}$ e per 65 valori di t uniformemente distribuiti sull'intervallo $(0,16)$, tenendo traccia dell'errore ρ ottenuto. In ognuno dei casi esaminati l'errore calcolato è stato dell'ordine di $6u$, questo evidenzia l'accuratezza dei due metodi anche nel caso in cui la matrice sia mal condizionata.

Test 4

Data la matrice $A = MDM^{-1}$ con $D = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{pmatrix}$ e $M = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ -3 & -2 & 1 \end{pmatrix}$ calcoliamo $A^{q/p}$ per alcuni valori fissati di p, al variare di $q \in \{1, \dots p-1\}$. Riportiamo i grafici di confronto tra gli errori ottenuti utilizzando il metodo di Iannazzo-Manasse e di Schur-Padé. Come misura dell'errore utilizziamo questa volta il rapporto $r = \dfrac{\|\tilde{Y} - MD^{q/p}M^{-1}\|}{\|MD^{q/p}M^{-1}\|}$.

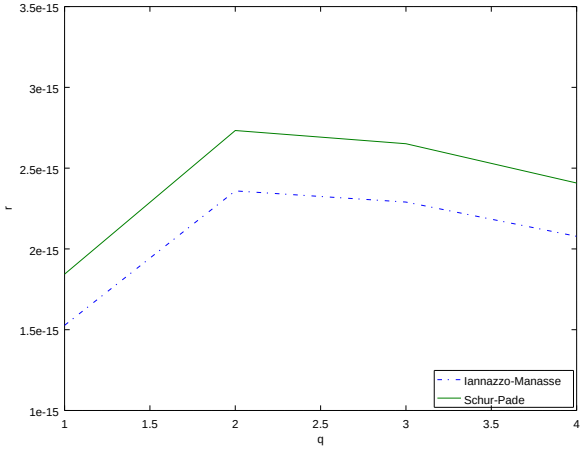

Figura 3.5: p=5

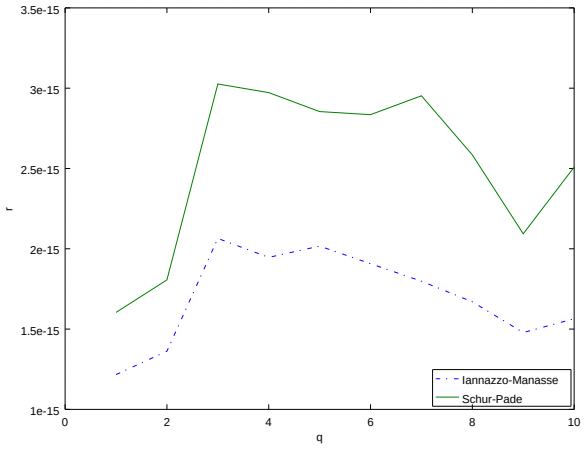

Figura 3.6: p=11

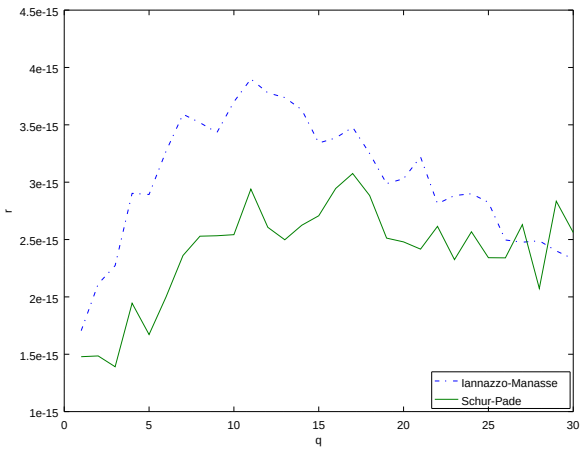

Figura 3.7: p=31

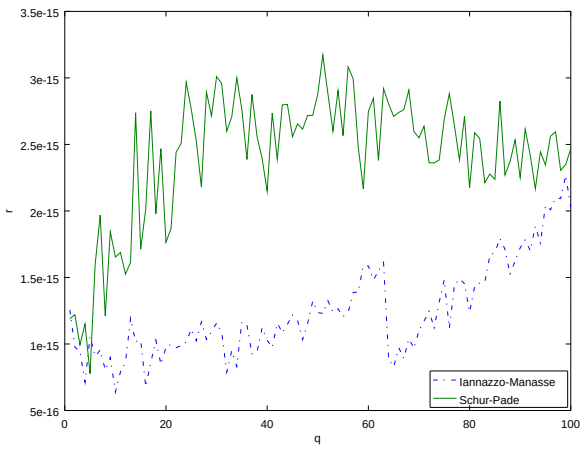

Figura 3.8: p=101

Test 5

Abbiamo visto che l'algoritmo di Iannazzo-Manasse può essere impiegato per calcolare le determinazioni non principali della radice p-esima. In questo test abbiamo calcolato le radici terze della matrice A=$\begin{pmatrix} a & 1 \\ 0 & b \end{pmatrix}$ variando le determinazioni scelte per ciascuno dei blocchi diagonali. Indicando con k e h la determinazione scelta rispettivamente per il primo e il secondo autovalore, nel passo corrispondente alla riga 3 dell'algoritmo 1 abbiamo preso $f(a) = \omega^{k-1}a^{1/3}$ e $f(b) = \omega^{h-1}b^{1/3}$ con $\omega = e^{2\pi \mathrm{i}/3}$.

Nelle tabelle sottostanti riportiamo i valori di

$$\rho_{kh} = \frac{\|\tilde{Y}^p - A\|}{\|\tilde{Y}\| \sum_{j=0}^{p-1} (\tilde{Y}^{p-1-j})^T \otimes \tilde{Y}^j\|} \qquad R_{kh} = \frac{\|\tilde{Y}^p - A\|}{\|A\|} \qquad \beta_{kh} = \frac{\|\tilde{Y}\|^p}{\|A\|}$$

al variare di tutte le combinazioni di $h, k \in \{1, 2, 3\}$ per $a = 1$, $b = 2$ e $b = 1 + 10^{-8}$ rispettivamente nella tabella 3.9 e 3.10. Osserviamo che in entrambi i casi quando la determinazione scelta è la stessa sui due autovalori l'indice β è prossimo a 1, ad indicare la stabilità dell'algoritmo, e che tale indice cresce prendendo determinazioni diverse. In particolare nel secondo caso, quando a e b sono vicini, l'indice β è dell'ordine di 10^{24}: questo sottolinea il cattivo condizionamento della radice così calcolata. In quest'ultimo caso, inoltre, ρ non è più un indice di accuratezza significativo.

$k \backslash h$	1	2	3
1	R =4.85e-17 ρ =1.67e-17 β =1.01	R =7.39e-16 ρ =6.77e-17 β =6.71	R =1.06e-15 ρ =9.72e-17 β =6.71
2	R =3.10e-16 ρ =2.84e-17 β =6.71	R =3.10e-16 ρ =2.8e-17 β =1.01	R =1.09e-15 ρ =9.95e-17 β =6.71
3	R =5.55-16 ρ =5.08e-17 β =6.71	R =7.36e-16 ρ =6.74e-17 β =6.71	R =1.06e-15 ρ =3.66e-16 β =1.00

Figura 3.9: $a = 1$, $b = 2$

$k \backslash h$	1	2	3
1	R =1.53e-16 ρ =5.53e-17 β =1.02	R =2.89e-09 ρ =5.57e-34 β =3.21e+24	R =8.03e-09 ρ =1.54e-33 β =3.21e+24
2	R =1.04e-08 ρ =2.01e-33 β =3.21e+24	R =4.94e-16 ρ =1.78e-16 β =1.00	R =8.14e-09 ρ =1.57e-33 β =3.21e+24
3	R =1.33e-08 ρ =2.57e-33 β =3.2e+24	R =5.38e-09 ρ =1.03e-33 β =3.21e+24	R =8.45e-16 ρ =3.05e-16 β =1.01

Figura 3.10: $a = 1$, $b = 1 + 10^{-8}$

Bibliografia

[1] George A. Baker. *Essenentials of Padé Approximants*. Academic Press, 1975

[2] Wilfrid N. Bailey. *Generalized Hypergeometric Series*. Cambridge University Press, 1935

[3] Åke Björck, Sven Hammarling, *A Schur Method for the Square Root of a Matrix*. Linear Algebra and its Applications 52/53, 1983

[4] Chun-Hua Guo, Nicholas J. Higham. *A Schur-Newton Method for the Matrix pth Root and its Inverse*. SIAM Journal on Matrix Analysis and Applications 28:3, 2006

[5] Federico Greco, Bruno Iannazzo. *A Binary Powering Schur Algorithm for Computing Primary Matrix Roots*. Numerical Algorithms 55, 2010

[6] Nicholas J. Higham, Lijing Lin. *A Schur-Padé Algorithm for Fractional Powers of a Matrix*. SIAM Journal on Matrix Analysis and Applications 32:3, 2011

[7] Nicholas J. Higham, Lijing Lin. *An Improved Schur-Padé Algorithm for Fractional Powers of a Matrix and Their Fréchet Derivatives*. SIAM Journal on Matrix Analysis and Applications 34:3, 2013

[8] Nicholas J. Higham. *Computing Real Square Root of a real Matrix*. Linear Algebra and its Applications 88/89, 1987

[9] Nicholas J. Higham. *Functions of Matrices: Theory and Computation*. SIAM, 2008

[10] Bruno Iannazzo, Carlo Manasse. *A Schur Logarithmic Algorithm for Fractional Power of Matrices*. SIAM Journal on Matrix Analysis and Applications 34:2, 2013

[11] Charles Kenney, Alan J. Laub *Padé error estimates for the logarithm of a matrix*. International Journal of Control 50:3, 1989

[12] Matthew I. Smith. *A Schur Algorithm for Computing Matrix pth Roots*. SIAM Journal on Matrix Analysis and Applications 24:4, 2003