Metodi di approssimazione

Mele Giampaolo

27 aprile 2012

Premessa

Questi appunti sono stati presi durante le lezioni di *Metodi di approssimazione* e sono solo le lezioni tenute dal professore, non sono inclusi i seminari degli studenti. Il principale motivo per cui li ho scritti è come esercizio di latex e per rivedere quello che si è fatto a lezione, quindi dubito che li rileggerò per correggere eventuali errori e si anticipa che l'intento di questi appunti non è una trattazione sistematica di tutti gli argomenti ma una presentazione quasi discorsiva per dare l'idea dei vari argomenti lasciando molti aspetti in sospeso. Invito comunque a segnalare eventuali errori e chiunque volesse il sorgente per ampliare/correggere questi appunti può mandarmi una mail a *mele@mail.dm.unipi.it*.

Mele Giampaolo

Indice

Capitolo 1

Problemi di calcolo di autovalori classici e generalizzati

Si affronteranno in questo capitolo i problemi agli autovalori classici con le eventuali generalizzazioni che nasceranno in modo naturale da esempi ed applicazioni che saranno mostrate nelle sezioni successive. Il punto fondamentale sarà la ricerca e lo studio delle strutture che si potranno sfruttare ai fini della risoluzione di tali problemi.

1.1 Esempi e applicazioni

1.1.1 Equazioni algebriche

Un problema di calcolo degli autovalori nasce in modo naturale dal problema del calcolo delle radici di un polinomio (risoluzione di un'equazione algebrica), ovvero dato

$$p(z) = p_0 + p_1 z + \cdots + p_n \in \mathbb{C}[z]$$

Si cercano le soluzioni di

$$p(z) = 0$$

L'approccio classico usato ad esempio in matlab è quello di costruire una matrice F tale

$$\det(F - zI) = \pm p(z)$$

E quindi il calcolo delle radici del polinomio equivale a calcolare gli autovalori di F. Questa matrice può esser scelta in tanti modi diversi, la scelta naturale è la forma *companion*, ovvero

$$F = \begin{pmatrix} & & -p_0/p_n \\ 1 & & -p_1/p_n \\ & \ddots & \vdots \\ & 1 & -p_{n-1}/p_n \end{pmatrix}$$

Osservazione 1.1. • F è in forma di Hessenberg. Si ricorda che nel momento in cui si vuogliono calcolare gli autovalori di una matrice, il primo passo è il calcolo della forma di Hessenberg, quindi si è risparmiato un passo.

- La matrice è sparsa, ma tale proprietà si perde durante l'esecuzione degli algoritmi noti (ad esempio il QR)
- $F = Q + uv^T$ con Q unitaria (la rappresentazione non è unica).
  Ad esempio
  $$Q = \begin{pmatrix} & & & 1 \\ 1 & & & \\ & \ddots & & \\ & & 1 & \end{pmatrix} \qquad u = \begin{pmatrix} -p_0/p_n - 1 \\ -p_1/p_n \\ \vdots \\ -p_{n-1}/p_n \end{pmatrix} \qquad v = e_n$$
  Quindi è possibile esprimere una matrice *companion* come una correzione di rango 1 di una matrice unitaria.
- Il problema è che durante l'algoritmo QR la matrice si riempe e in generale il costo è $O(n^3)$, daltronde dato che si hanno n dati in input ed n in output il costo sembra eccessivo. Quindi l'obbiettivo sarà fare delle osservazioni sull'algoritmo QR per abbassare il costo computazionale.

E' noto che nel momento in cui una matrice viene rappresentata da un computer si avrà un errore di rappresentazione

$$F = \widehat{F} + \Delta F$$

In generale

$$\widehat{f_{i,j}} = f_{i,j}(1 + \epsilon_{i,j}) = f_{i,j} + f_{i,j}\epsilon_{i,j} \qquad |\epsilon_{i,j}| \leq u \quad u \text{ è la precisione di macchina}$$

Dal teorema di Bauer-Fike (inserire riferimento)

$$\|\Delta F\| \leq u\|F\|$$

Quindi si vuole che la norma di F non sia troppo grande, nel caso della *companion* si avranno problemi quando p_n è troppo piccolo. Assumendo $p_n \neq 0$, è possibile fattorizzare

$$F = \begin{pmatrix} & & -p_0/p_n \\ 1 & & -p_1/p_n \\ & \ddots & \vdots \\ & 1 & -p_{n-1}/p_n \end{pmatrix} = \begin{pmatrix} & & -p_0 \\ 1 & & -p_1 \\ & \ddots & \vdots \\ & 1 & -p_{n-1} \end{pmatrix} \begin{pmatrix} 1 & & \\ & \ddots & \\ & & p_n \end{pmatrix}^{-1} = \widehat{F}E$$

Dunque calcolare il calcolo degli autovalori $Fw = \lambda w$ equivale a

$$\widehat{F}E^{-1}w = \lambda w \iff \widehat{F}(E^{-1}w) = \lambda E(E^{-1}w)$$

Quindi posto $z = E^{-1}w$ si ha il problema

$$\widehat{F}z = \lambda Ez$$

In questo caso non è stato necessario fare divisioni, quindi anche se p_n è piccolo comunque non è stato necessario dividere. Daltronde si è riformulato il problema, questo non è più un problema classico agli autovalori (chiamato anche *EP*: eigenvalue problem) ma un problema generalizzato agli autovalori (chiamato *GEP*: generalized eigenvalue problem).

- $(\widehat{F}, E)$ è detta *matrix pair*
- $\widehat{F} - \lambda E$ è detta *matrix pancil*

Si osserva inoltre che

- $\widehat{F}$ è in forma di Hessenberg
- $\widehat{F} = Q + uv^T$
- $E = I + \dfrac{e_n e_n^T}{p_n}$

L'equivalente dell'algoritmo *QR* che si presenterà a breve, lavorerà sulla *matrix pair*.

1.1.2 Polynomial eigenvalue problem (PEP)

Si consideri ora il problema: date le matrici $A_1, \ldots, A_n$, si vuole determinare λ tale che

$$\det\left(\sum_{i=1}^{n} A_i \lambda^i\right) = 0$$

Ovvero si cercano gli scalari λ tali che esiste un vettore non nullo v tale che

$$\left(\sum_{i=1}^{n} A_i \lambda^i\right) v = 0$$

E' possibile pensare al *PEP* come una generalizzazione del *GEP*. Volendo rappresentare le varie generalizzazioni che sin son fatte sin ora

$$EP \rightarrow GEP \rightarrow PEP$$

Per risolvere il *PEP* ci si ricondurrà al caso appena esposto, quindi si cercherà di costruire una matrice *companion*, si supponga $A \in \mathbb{C}^{k \times k}$, allora

$$\widehat{F} = \begin{pmatrix} & & & -A_0 \\ I_k & & & -A_1 \\ & \ddots & & \vdots \\ & & I_k & -A_{n-1} \end{pmatrix} \qquad E = \begin{pmatrix} I_k & & & \\ & \ddots & & \\ & & I_k & \\ & & & A_n \end{pmatrix}$$

Quindi ci si riesce a ricondurre al caso del *GEP* seppur avendo una struttura a blocchi. Qui si riesce dunque a non invertire A_n (che potrebbe esser malcondizionata), meglio ancora, A_n potrebbe non esser invertibile, ma il problema continua ad avere senso.

1.1.3 Metodi divide et impera

Per il calcolo degli autovalori c'è anche un algoritmo del tipo *divide et impera*, un aspetto negativo di questo approccio è la necessità di calcolare autovalori e autovettore di una matrice della forma

$$A = D + \theta u u^H$$

con D diagonale ed Hermitiana. Per applicare *QR* è necessario

- Portare la matrice in forma tridiagonale $A \rightarrow T$

- Applicare a T l'algoritmo QR e quindi un costo di $O(n^2)$

Il problema è dunque il calcolo di T. Portare in forma tridiagonale A ha costo $O(n^3)$ se si utilizza Householder, se si utilizza Lanczos il costo in questo esempio è $O(n^2)$.

Osservazione 1.2. Il metodo di Lanczos costa $O(n)$ per passo dato che si fa un prodotto matrice-vettore che in questo esempio costa appunto $O(n)$, ma il problema è che non si ha stabilità numerica.

Quindi ci si chiede se è possibile, utilizzando un algoritmo numericamente stabile, calcolare T.
(INSERIRE UNA DESCRIZIONE DI LANCZOS)

1.1.4 Basi di polinomi ed equazioni algebriche

La classica equazione algebrica è

$$\sum_{i=0}^{n} p_i z^i = 0$$

Implicitamente si è scelta come base dello spazio dei polinomi quella dei monomi. Una base differente è la seguente

$$\phi_i(z) = z^i(1-z)^{n-i}$$

quindi una equazione algebrica in questa base si esprime come

$$\sum_{i=0}^{n} p_i \phi_i(z) = 0$$

In altri contesti si usano basi di polinomi che presentano il fenomeno di ricorrenza a tre termini

$$\begin{cases} \phi_{i+1}(z) = (z-\alpha_1)\phi_i(z) + \gamma_i \phi_{i-1}(z) & 1 \leq i \leq n-1 \\ \phi_0(z) = 1 \\ \phi_{-1}(z) = 0 \end{cases}$$

Ad esempio i polinomi caratteristici di una sottomatrice principale di una matrice tridiagonale rispettano tale condizione.

Supponiamo $\gamma_i > 0$, allora è possibile scrivere per comodità γ_i^2 al posto di γ_i. In tal caso il polinomio $p(z)$ lo si può esprimere come polinomio caratteristico della matrice seguente

$$C = \begin{pmatrix} \alpha_1 & \gamma_1 & & & \\ \gamma_1 & \alpha_2 & \gamma_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \gamma_{n-2} & \alpha_{n-1} & \gamma_{n-1} \\ & & & \gamma_{n-1} & \alpha_n \end{pmatrix} + ue_n^T$$

In questi casi si ha una matrice in forma di Hessenberg

$$H = T + ue_n^T$$

Dove T è triangolare e simmetrica. Quindi ci si ripropone il problema dell'utilizzo del metodo QR.

1.1.5 Base di Lagrange

Quando si approssimano le funzioni continue mediante l'interpolazione polinomiale si ha

$$f(x) =\simeq p(x) = \sum_{i=0}^{n} f(x_i) L_i(x)$$

Dove

$$L_i(x) = \prod_{\substack{j=0 \\ j \neq i}} \frac{x - x_j}{x_i - x_j}$$

Questi sono detti polinomi di Lagrange, quindi il polinomio di interpolazione sarà

$$p(x) = \sum_{i=0}^{n} \frac{f(x_i)}{\prod_{j=0, j\neq i} (x_i - x_j)} \prod_{\substack{j=0 \\ j \neq i}} (x - x_j)$$

Si definisce

$$\omega(x) = \prod_{j=0}^{n} (x - x_j)$$

quindi si ottiene

$$p(x) = \sum_{i=0}^{n} \frac{f(x_i)}{\omega'(x_i)} \prod_{\substack{j=0 \\ j \neq i}} (x - x_j)$$

Si supponga di avere

$$p(z) = p_0 + p_1 z + \cdots + p_{n-1} z^{n-1} + z^n$$

si prenderanno n nodi $x_1 < \cdots < x_n$, quindi

$$g(z) = p(z) - \prod_{i=1}^{n} (z - x_i) = \sum_{i=0}^{n} \frac{g(x_i)}{\omega'(x_i)} \prod_{\substack{j=0 \\ j \neq i}} (z - x_j) = \sum_{i=0}^{n} \frac{p(x_i)}{\omega'(x_i)} \prod_{\substack{j=0 \\ j \neq i}} (z - x_j)$$

dove si è usato $g(x_i) = p(x_i)$. Quindi

$$p(z) = \prod_{i=1}^{n} (z - x_i) + \sum_{i=1}^{n} \frac{p(x_i)}{\omega'(x_i)} \prod_{\substack{j=0 \\ j \neq i}} (z - x_j)$$

Si ricorda che

$$\begin{aligned} \det(zI - D - uv^T) &= \det(zI - D) \det(I - (zI - D)^{-1} uv^T) \\ &= \prod_{j=1}^{n} (z - d_j)(1 + v^T (zI - D)^{-1} u) \\ &= \prod_{j=1}^{n} (z - \alpha_j) \left(1 + \sum_{i=1}^{n} \frac{v_i u_i}{2 - \alpha_i} \right) \end{aligned}$$

In questo caso si prenderanno

- $d_i = x_i$
- $u_i = p(x_i)$

- $v_i = \dfrac{1}{\omega'(x_i)}$

Quindi nuovamente per calcolare gli zeri di una equazione algebrica sarà necessario trovare gli autovalori di una matrice della forma $D + uv^T$ e questa ha anche una forma di matrice *companion*. Quindi si è ricondotto il calcolo di un polinomio di interpolazione a quello degli autovalori di una matrice. Osserviamo che a priori D non è ne unitaria ne Hermitiana. Se si vuole che D sia unitaria bisognerà scegliere i nodi sulla circonferenza unitaria, se invece la si vuole Hermitiana i nodi devono essere sull'asse reale.

1.2 Algoritmo QR

Segue un breve richiamo sull'algoritmo QR per il calcolo degli autovalori

1.2.1 Versione base

Si supponga di voler calcolare lo spettro di una matrice A, allora l'idea è di fattorizzare A come prodotto di una matrice unitaria per una matrice triangolare destra $A = QR$, considerare la matrice $A_1 = RQ$ e nuovamente fattorizzare A_1 come prodotto di una matrice unitaria per una matrice triangolare destra e così via, ovvero generare una successione che parte da $A_0 = A$ e poi

$$\begin{cases} A_k = Q_k R_k \\ A_{k+1} = R_k Q_k \end{cases} \qquad k \geq 0$$

Osservazione 1.3. La fattorizzazione QR non è unica, quindi la successione non è univocamente determinata.

Osservazione 1.4. Ogni matrice A_k della successione è unitariamente simile alla matrice A_0, infatti

$$A_{k+1} = R_k Q_k = Q_k^H Q_k R_k Q_k = Q_k^H A_k Q_k$$

Da questa relazione segue che A_{k+1} è unitariamente simile ad A_k, quindi essendo questa una relazione di equivalenza A_{k+1} è unitariamente simile ad A_0.

Osservazione 1.5. Se A è invertibile, allora A_k è invertibile e quindi R_k è invertibile, allora

$$A_{k+1} = R_k Q_k = R_k Q_k R_k R_k^{-1} = R_k A_k R_k^{-1}$$

Da questo segue che

- Se A_0 è di Hessenberg allora le A_k sono di Hessenberg
- Se A_0 è tridiagonale Hermitiana allora A_k è tridiagonale Hermitiana

1.2.2 Algoritmo QR con shift

Il problema dell'algoritmo QR è la lentezza, infatti nella sua versione base la convergenza è molto lenta, quindi le implementazioni che si considerano sono le varianti con shift, ovvero si considera la successione

$$\begin{cases} A - \alpha_k I = Q : kR_k \\ A_{k+1} := R_k Q_k + \alpha_k I \end{cases}$$

Osservazione 1.6. Si può mostrare che nella versione con shift dell'algoritmo QR valgono le stesse proprietà dell'algoritmo di base, cioè è conservata la struttura di Hessenberg, la struttura tridiagonale, le matrici generate sono unitariamente simili alla matrice iniziale.

Il vantaggio è che se si scelgono in modo opportuno gli α_n (sono detti shift) la convergenza è rapida. Questo algoritmo più esser scritto in forma più compatta, ponendo $p_k(z) = z - \alpha_k$ si ha

$$\begin{aligned} p_k(A_k) &= Q_k R_k \\ \\ A_{k+1} &= Q_k^H Q_k (R_k Q_k + \alpha_k I) \\ &= Q_k^H (A_k - \alpha_k I) Q_k + \alpha_k I \\ &= Q_k^H A_k Q_k \end{aligned}$$

Ovvero si è ottenuto

$$\begin{cases} p_k(A_k) = Q_k R_k \\ A_{k+1} = Q_k^H A_k Q_k \end{cases}$$

In questo caso si ha appunto che p_k è un polinomio di primo grado, nasce spontaneamente la generalizzazione, si può infatti pensare di prendere un polinomio di grado più alto. In questo caso si parla di algoritmo QR con shift singolo.

Esempio 1.2.1. In quale contesto può esser utile considerare un polinomio di grado più elevato? Ad esempio se la matrice A è ad elementi reali e si vogliono approssimare gli autovalori rimanendo in aritmetica reale. L'idea è di shiftare per un α, ma A è una matrice reale ed α non lo è ma è un'approssimazione di un autovalore, allora bisogna assumere che ci sia anche $\bar{\alpha}$ nello spettro della matrice, quindi bisognerebbe fare prima uno shift con α e poi uno shift con $\bar{\alpha}$. A questo punto si osserva che fare due passi dell'algoritmo usando prima α e poi $\bar{\alpha}$ equivale a fare un passo dell'algoritmo QR scegliendo come polinomio $p_k(z) = (z - \alpha)(z - \bar{\alpha})$, questo è un polinomio a coefficienti reali, quindi si resta in aritmetica reale. In questo contesto si parla di shift quadratico.

Si può mostrare che, assundo che l'algoritmo converga in un numero lineare di passi (congettura), si può implementare l'algoritmo con un costo $O(n^2)$ per passo, quindi il costo totale è $O(n^3)$. Si supponga ora A_k sia in forma di Hessenberg, allora

$$p_k(z) = z^2 + a_k z + b_k \qquad a_k, b_k \in \mathbb{R}$$

Quando si valuta il polinomio nella matrice

$$p_k(A_k) = A_k^2 + a_k A_k + b_k \qquad a_k, b_k \in \mathbb{R}$$

Di questa matrice si dovrà poi fare la fattorizzazione QR. Il problema è che se A è di Hessenberg (in particolare ha la sottodiagonale principale piena), allora A^2 ha le prime due sottodiagonali principali piene, quindi al crescere di A_k si ha un fenomeno di riempimento. Inoltre moltiplicare una matrice in forma di Hessenberg per una matrice in forma di Hessenberg ha costo cubico. Questi problemi si risolvono introducendo il metodo QR implicito.

1.2.3 Algoritmo QR implcito

Si affronta ora la variante del metodo QR che è attualmente implementata in svariati software quali ad esempio matlab Questa variante è nota come algoritmo QR con shift implicito. Si riprenda la versione con shift presentata prima

$$\begin{cases} p_k(A_k) = Q_k R_k \\ A_{k+1} = Q_k^H A_k Q_k \end{cases}$$

Dove per comodità supponiamo di fare uno shift singolo (non è difficile estendere al caso quadratico), quindi p_k è un polinomio di primo grado. Per semplicità consideriamo solo un passo dell'algoritmo, quindi si elimina l'indice k e più sinteticamente si considera il problema

$$\begin{cases} p(A_k) = QR \\ A_1 = Q^H A Q \end{cases} \qquad p(z) = z - \alpha$$

Pertato è necessario calcolare una rattorizzazione QR della seguente matrice

$$A - \alpha I = \begin{pmatrix} a_{1,1} - \alpha & \dots & \dots & \dots & \\ \beta_1 & a_{2,2} - \alpha & & & \vdots \\ & \ddots & \ddots & & \vdots \\ & & \ddots & \ddots & \vdots \\ & & & \beta_{n-1} & a_{n,n} - \alpha \end{pmatrix}$$

Si ricorda che nel caso che si sta considerando si suppone di aver già portato A in forma di Hessenberg, quindi chiaramente anche $A - \alpha I$ è di Hessenberg. Ora quindi bisogna trovare una fattorizzazione QR di questa matrice, quindi come primo passo bisognerà trovare la matrice di Householder P_1 tale per cui $P_1(A - \alpha I)$ ha nella prima colonna un multiplo di e_1 (primo vettore della base canonica).

Osservazione 1.7. Se $v \in \mathbb{R}^n$ ha solo le prime k coordinate non nulle, allora la matrice di Householder che lo trasfrorma in un multiplo della base canonica sarà

$$P = I - 2vv^h = \left(\begin{array}{c|c} G & \\ \hline & I_{n-k} \end{array}\right)$$

Dove G è una matrice di ordine k unitaria.

Usando l'osservazione si ha che

$$P_1 = \left(\begin{array}{c|c} G_1 & \\ \hline & I_{n-2} \end{array}\right)$$

A questo punto è chiaro come continuare, si calcola la matrice di Householder che rende il secondo vettore colonna (considerato dalla seconda coordinata in poi) di $A - \alpha I$ multiplo del primo vettore della base canonica (si usa il solito algoritmo di fattorizzazione QR) è importante l'osservazione per capire come sono fatte le varie matrici di Householder, quindi si avrà

$$\left(\begin{array}{c|c} I_{n-2} & \\ \hline & G_{n-1} \end{array}\right) \dots \left(\begin{array}{c|c|c} 1 & & \\ \hline & G_2 & \\ \hline & & I_{n-3} \end{array}\right) \left(\begin{array}{c|c} G_1 & \\ \hline & I_{n-2} \end{array}\right) (A - \alpha I) = R$$

Dove appunto le G_i sono matrici di ordine 2 unitarie, quindi si avrà

$$Q^H = \left(\begin{array}{c|c} I_{n-2} & \\ \hline & G_{n-1} \end{array}\right) \dots \left(\begin{array}{c|c|c} 1 & & \\ \hline & G_2 & \\ \hline & & I_{n-3} \end{array}\right) \left(\begin{array}{c|c} G_1 & \\ \hline & I_{n-2} \end{array}\right)$$

Quindi il primo passo del metodo QR consisterà nel calcolare A_1 come

$$\begin{aligned} A_1 \quad &= Q^H A Q \\ &\left(\begin{array}{c|c} I_{n-2} & \\ \hline & G_{n-1} \end{array}\right) \dots \left(\begin{array}{c|c|c} 1 & & \\ \hline & G_2 & \\ \hline & & I_{n-3} \end{array}\right) \left(\begin{array}{c|c} G_1 & \\ \hline & I_{n-2} \end{array}\right) \\ &A \left(\begin{array}{c|c} G_1 & \\ \hline & I_{n-2} \end{array}\right) \left(\begin{array}{c|c|c} 1 & & \\ \hline & G_2 & \\ \hline & & I_{n-3} \end{array}\right) \dots \left(\begin{array}{c|c} I_{n-2} & \\ \hline & G_{n-1} \end{array}\right) \end{aligned}$$

L'idea è di non svolgere i vari prodotti per calcolare Q e poi A_1, ma di sfruttare la simmetria del problema, quindi si pone

$$A^{(1)} = \left(\begin{array}{c|c} G_1 & \\ \hline & I_{n-2} \end{array}\right) A \left(\begin{array}{c|c} G_1 & \\ \hline & I_{n-2} \end{array}\right)$$

Si osserva subito che $A^{(1)}$ è unitariamente simile ad A, inoltre si definiscono in modo analogo $A^{(i)}$ nel seguente modo

$$\begin{cases} A^{(i+1)} = P_i A^{(i)} P_i \\ A^{(0)} = A \end{cases}$$

Dove

$$P_i = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & G_i & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}$$

Si osserva subito che tutte le $A^{(i)}$ sono unitariamente simili ad A, per semplicità si analizza solo $A^{(1)}$ ma in modo ovvio si estende a tutte le $A^{(i)}$.

E' chiaro che $A^{(1)}$ avrà la forma

$$A^{(1)} = \begin{pmatrix} * & * & * & * & * & \dots \\ * & * & * & * & * & \dots \\ \bullet & * & * & * & * & \dots \\ & & * & * & * & \dots \\ & & & * & * & \dots \\ & & & & \ddots & \end{pmatrix}$$

Quindi $A^{(1)}$ è quasi in forma di Hessenberg, nel senso che l'unico difetto è l'elemento che è stato indicato con $\bullet$, tale elemento è noto anche come *bulge* (tradotto dall'inglese è *bernoccolo*), quindi bisognerà eliminarlo utilizzando un algoritmo di *bulge cleansing*, si seguirà la stessa idea mostrata prima con le matrici di Huseholder per eliminare il *bulge*.

Esempio 1.2.2 (Algoritmo di *bulge cleansing*)**.** Si mostra ora un esempio di *bulge cleansing* con una matrice 5×5, per semplicità riempiamo la matrice di soli simboli e non di numeri

$$A = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ \bullet & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{pmatrix}$$

Per eliminare il *bulge* si utilizzano al solito le matrici di Householder e si ripete lo stesso ragionamento fatto prima delle matrici 2×2 che in questo caso verranno chiamate $\widehat{G_i}$, quindi

$$\left(\begin{array}{c|c|c} 1 & & \\ \hline & \widehat{G_2} & \\ \hline & & I_2 \end{array}\right) \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ \bullet & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{pmatrix} \left(\begin{array}{c|c|c} 1 & & \\ \hline & \widehat{G_2} & \\ \hline & & I_2 \end{array}\right) = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & \bullet & * & * & * \\ & & & * & * \end{pmatrix}$$

Quindi il *bulge* è sceso di una riga, ripetendo l'algoritmo

$$\left(\begin{array}{c|c|c} I_2 & & \\ \hline & \widehat{G_3} & \\ \hline & & 1 \end{array}\right) \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & \bullet & * & * & * \\ & & & * & * \end{pmatrix} \left(\begin{array}{c|c|c} I_2 & & \\ \hline & \widehat{G_3} & \\ \hline & & 1 \end{array}\right) = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & \bullet & * & * \end{pmatrix}$$

Ripetendo un'ultima volta

$$\left(\begin{array}{c|c} I_3 & \\ \hline & \widehat{G_4} \end{array}\right) \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & \bullet & * & * \end{pmatrix} \left(\begin{array}{c|c} I_3 & \\ \hline & \widehat{G_4} \end{array}\right) = \begin{pmatrix} * & * & * & * & * \\ * & * & * & * & * \\ & * & * & * & * \\ & & * & * & * \\ & & & * & * \end{pmatrix}$$

Da questo semplice esempio è chiaro come agisce l'algoritmo di *bulge cleansing* ed è chiaro che il suo costo computazionale è basso, infatti in realtà i prodotti tra matrici sono ogni volta combinazioni di due righe e due colonne, quindi per eliminare un *bulge* usando matrici unitarie il costo sarà $O(n)$.

In definitiva l'idea è di calcolare $A^{(1)}$ che avrà un *bulge* ed utilizzare il *bulge cleansing* per rendere $A^{(1)}$ di Hessenberg. Si definisce $\widehat{A_1}$ la matrice ottenuta da $A^{(1)}$ dopo il *bulge cleansing*. In generale A_1 sarà diversa da $A^{(1)}$ ma vale il seguente teorema.

Teorema 1.2.1 (Implicit QR theorem)**.** Esiste una matrice di fase (cioè diagonale e unitaria) S tale che

$$A_1 = S\widehat{A_1}S^H$$

Quindi per calcolare A_1 (a meno di matrici di fase) è sufficiente conoscere la prima colonna di $p_k(A)$, è dunque sufficiente calcolare $p_k(A)e_1$ e questo lo si calcola in $O(n^2)$, dalla conoscenza di $p_k(A)e_1$ si ricava P_1, si calcola $A^{(1)}$ e con il *bulge cleansing* si trova $\widehat{A_1}$ che sarà a meno di una matrice di fase A_1, quindi si itera l'algoritmo fino a convergenza. Anche se non esiste una dimostrazione puramente matematica si trova che l'algoritmo (con opportune varianti ed espedienti tecnici) converge in meno di $6n$ passi.

Purtroppo non esistono forti risultati di convergenza per l'algoritmo QR, solo sotto ipotesi molto forti si riesce ad ottenere un teorema di convergenza che è il seguente.

Teorema 1.2.2 (Convergenza del metodo QR)**.** Se A ha tutti autovalori con modulo differente, esiste una successione di matrici di fase tale che

$$S_k^H A_k S_k \to R$$

Dato che $S_k^H A_k S_k$ sono tutte unitariamente simili ad A, allora anche R (triangolare superiore) è unitariamente simile ad A, quindi sulla diagonale ci saranno gli autovalori di A.

Sperimentalmente si trova che in realtà questo algoritmo funziona sempre per ogni tipo di matrice a meno di qualche astuzia tecnica (shift variabile) o tecniche di restarting.

Osservazione 1.8. Saremo interessati a studiare l'algoritmo QR nel caso in cui A è una correzione di rango basso di una matrice unitaria o di una matrice Hermitiana, ovvero

$$A = Q + UV^H \qquad \text{con} \qquad \begin{array}{l} Q \in \mathbb{C}^{n\times n} \\ U, V \in \mathbb{C}^{n\times k} \\ k << n \end{array}$$

Dove appunto $QQ^H = Q^HQ = I$ oppure $Q = Q^H$. Cosa succede se si applica l'algoritmo QR ad A?

$$\begin{cases} A_0 = A \\ A_k = U_k A_0 U_k \end{cases}$$

Dove appunto $U_k^H U_k = U_k U_k^H = I$, quindi A_k sono tutte unitariamente simili ad A, ma allora

$$\begin{aligned} A_k &= U_k^H (Q + UV^H) U_k^H \\ &= U_k^H Q U_k + U_k^H U V^H U_k^H \\ &= U_k^H Q U_k + U^{(k)} V^{(k)H} \end{aligned}$$

Dove si è posto

$$\begin{aligned} U^{(k)} &:= U_k^H U \\ V^{(k)H} &:= V^H U_k^H \end{aligned}$$

Quindi le matrici A_k sono anch'esse correzioni di rango basso di matrici unitarie o Hermitiane.

Cosa si può dire di una matrice in forma di Hessenberg che sia esprimibile come una correzione di rango basso di una matrice unitaria o Hermitiana? Lo si vedrà più avanti.

Osservazione 1.9. Quanto visto nell'ossevazione precedente non è valido se consideriamo correzioni di rango basso di una matrice normale. Se ad esempio si considera

$$A = A_0 = D + uv^H$$

Dove D è diagonale. In questo caso non si riesce a dimostrare nessuna proprietà strutturale dell'algoritmo QR per questa matrice. Questo non vuol dire che non ce ne siano ma che attualmente non sono note e non ci sono lavori che affrontano tali problemi.

Volendo contestualizzare, in realtà qualcosa si può dire, ad esempio se supponiamo che gli elementi di D si trovino su una circonferenza o su una conica o sull'asse reale ...; in tali casi qualcosa si può dire ma computazionalmente non ha importanza fato che queste proprietà si perdono con i passi dell'algoritmo QR.

Osservazione 1.10. In generale l'essere normale è una proprietà che viene preservata dall'algoritmo QR, ovvero se A è normale, anche A_k sono normali. Daltronde è difficile trovare una matrice normale che non sia Hermitiana o unitaria.

Si può mostrare che A è normale se e soltanto se vale

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

Quindi è così possibile determinare una matrice normale non Hermitiana, non unitaria e non diagonale. Daltronde la normalità viene persa computazionalmente durante l'esecuzione a causa degli errori, questo fenomeno è noto come *allontanamento dalla normalità*. La normalità è una proprietà che nel caso generale è difficile da mantenere computazionalmente a meno di ricadere nei casi di matrici Hermitiane, unitarie o diagonali.

1.3 Algoritmo QR per il GEP

Si vuole risolvere il GEP (generalized eigenvalue problem)

$$Ax = \lambda Bx$$

Ovvero trocare i λ senza però invertire B.

1.3.1 Caso con A simmetrica e B definita positiva

Prima idea

Dal teorema di diagonalizzazione simultanea si ha che

$$\begin{aligned} B &= QDQ^T \\ Q^TBQ &= D = D^{1/2}D^{1/2} \\ (D^{-1/2}Q^T)B(QD^{-1/2}) &= I \end{aligned}$$

Quindi a meno di cambiare base B è la matrice identità, quindi si può risolvere il GEP

$$\begin{aligned} &Ax = \lambda Bx \\ &(D^{-1/2}Q^T)A(QD^{-1/2})x = \lambda(D^{-1/2}Q^T)B(QD^{-1/2}) \\ &(D^{-1/2}Q^T)A(QD^{-1/2})x = \lambda I \end{aligned}$$

Quindi ci si è ricondotti alla risoluzione di un sistema lineare. Daltronde questo metodo è troppo costoso, richiede il calcolo di autovalori e autovettori per B quindi non efficiente dal punto di vista numerico.

Decomposizione di Cholesky

Si ricorda che matrice definita positiva B ha una decomposizione $B = LL^T$ dove L è triangolare, se si vuole l'unicità della decomposizione si deve imporre che L deve avere elementi positivi sulla diagonale principale. A questo punto

$$L^{-1}BL^{-T} = I$$

quindi

$$\begin{aligned} &Ax = \lambda Bx \\ &Ax = \lambda LL^Tx \\ &L^{-1}AL^{-T}(L^Tx) = \lambda(L^Tx) \end{aligned}$$

Ponendo $y := L^Tx$

$$(L^{-1}AL^{-T})y = \lambda y$$

Quindi ci si è ridotti al problema classico degli autovalori.

Questo non risolve tutti i problemi, infatti anche se non si inverte B comunque si inverte L, e se B è malcondizionata allora l'inversione di L può dar problemi.

1.3.2 Algoritmo *QZ*

Si vuole ora dare un algoritmo per il calcolo degli autovalori generalizzati del pancil (A, B) che prescinda dall'inversione di B. Nel 1973 Moler (proprietario di matlab) e Stewart trovano un algoritmo, noto come QZ che permette di risolvere il GEP senza l'invesione di B.

Quindi si opererà sulla coppia (A_0, B_0) e si considerano trasfrormazioni

$$(A_0, B_0) \to (A_1, B_1)$$

Dove

$$\begin{cases} A_1 = Q_1 A_0 U_1 \\ B_1 = Q_1 B_1 U_1 \end{cases}$$

Con Q_1 e U_1 sono ortogonali.

Quindi si ha

$$\begin{aligned} \det(A_1 - \lambda B_1) &= \det(Q_1 A_0 U_1 - \lambda Q_1 B_1 U_1) \\ &= \pm \det(A_0 - \lambda B_0) \end{aligned}$$

Q_1 e U_1 devono esser determinate in modo tale che $B_1^{-1} A_1$ (purchè la si possa costruire) sia essenzialmente la matrice ottenuta dal metodo QR dopo il primo passo (per essenzialmente uguale si intende a meno di matrici di fase). Si può sempre supporre che A_0 sia di Hessenberg e B_0 triangolare (ci si può sempre ridurre a forme di questo tipo). Quindi si avrà

$$\begin{aligned} A_k &\to R \\ B_k &\to S \end{aligned}$$

Dove R e S sono matrici triangolari superiori. A questo punto si ha il problema più semplice

$$Rx = \lambda S x$$

Quindi si ha che gli autovalori sono i rapporti tra gli elementi diagonali, quindi

$$\lambda_i = \frac{r_{i,i}}{s_{i,i}}$$

Il problema è se S ha degli zeri sulla diagonale principale, ovvero se B è singolare. Questo però si può giustificare, infatti in questo caso il polinomio caratteristico generalizzato sarà

$$\begin{aligned} p(z) &= \det(A - \lambda B) \\ &= p_0 + p_1 z + \cdots + p_s z^s \end{aligned}$$

In generale $s \leq n$, quindi ci si deve aspettare s autovalori e non n. Ad esempio se B è la matrice nulla si averebbe un polinomio costante. Non è difficile mostrare che $s = n$ se e soltato se $\det B \neq 0$.

Quindi se B non è invertibile è possibile considerare il polinomio caratteristico reciproco

$$q(z) := z^n p\left(\frac{1}{z}\right)$$

Questo avrà grado più piccolo di n, qui si avranno di coseguenza degli autovalori nulli e questi autovalori nulli derivano da autovalori all'infinito per $p(z)$. Quindi è sensato dire che se B non è invertibile allora ci possono essere degli autovalori all'infinito.

Il problema diventa decisamente più complesso quando anche la matrice A è singolare, in questi casi si può avere che per un certo i vale $r_{i,i} = s_{i,i} = 0$ quindi si avrebbe $\lambda_i = 0/0$ quindi indeterminazione. In tal caso matlab avverte l'utente dicendogli che il problema è mal posto dato che non si ha continuità rispetto ai dati, piccole perturbazioni possono trasfrormare autovalori infiniti in autovalori nulli.

Osservazione 1.11. Se A o B sono correzioni di rango basso di matrici unitarie, questa struttura si mantiene con l'algoritmo QZ, la verifica è semplice. Ma se invece si ha la struttura Hermitiana, questa in generale non si conserva.

1.4 Matrici con struttura di rango

1.4.1 Definizioni ed esempi

In questa sottosezione saranno definite le matrici con struttura di rango, ma è preferibile arrivarci gradualmente iniziando con osservazioni ed esempi

Osservazione 1.12. Si consideri $A = H + uv^h$ una correzione di rango 1 di una matrice Hermitiana e si supponga A già in forma di Hessenberg (si è mostrato che l'algoritmo per porta in forma di Hessenberg conserva la struttura Hermitiana corretta di rango 1), allora

$$\begin{aligned} A - A^H &= H + uv^H - H^H - uv^H \\ &= uv^H - vu^H \end{aligned}$$

Quindi $A - A^H$ è una matrice di rango 2 antihermitiana. Ma allora essendo A in forma di Hessenberg

$$\begin{aligned} A &= A^H + (uv^H - vu^H) \\ &= \begin{pmatrix} * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & & * & * & * & * & * \\ & & & * & * & * & * \\ & & & & * & * & * \\ & & & & & * & * \end{pmatrix} - \begin{pmatrix} & & & & & & \\ & & & & & & \\ & & & \text{rk } 2 & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \\ & & & & & & \end{pmatrix} \\ &= \begin{pmatrix} * & * & & & & & \\ * & * & * & & & & \\ & * & * & * & & & \\ & & * & * & * & & \\ & & & * & * & * & \\ & & & & * & * & * \\ & & & & & * & * \end{pmatrix} \end{aligned}$$

Dove quello che si intende è che A è una matrice che ha la parte triangolare superiore dalla seconda diagonale come una matrice di rango 2.

Quindi si osserva che

$$\begin{aligned} \max_{1 \le k \le n-1} (\quad rkA(1:k, k+1:n) \quad) \le 3 \\ \max_{1 \le k \le n-1} (\quad rkA(k+1:n, 1:k) \quad) \le 1 \end{aligned}$$

E per quanto mostrato sin ora segue che le due proprietà appena mostrate continuano a valere durante ogni passo dell'algoritmo QR, infatti ad ogni passo la matrice A_k sarà sempre una correzione di rango 1 di una matrice Hermitiana.

Notazione 1.4.1. Con $A(p:q, r:s)$ si intende la sottomatrice di A che contiene le righe dalla p-esima alla q-esima e le colonne dalla r-esima alla s-esima esattamente come con la notazione di matlab.

D'ora in avanti si farà riferimento alle notazioni e alle definizioni introdotte da Eidelman e Gohberg.

Definizione 1.4.1. Una matrice A è quasi separabile superiorimente di ordine a se

$$\max_{1 \le k \le n-1} (\quad rkA(1:k, k+1:n) \quad) \le a$$

Definizione 1.4.2. Una matrice A è quasi separabile inferiormente di ordine b se

$$\max_{1\leq k\leq n-1} (\quad rkA(k+1:n, 1:k) \quad) \leq b$$

Quindi nell'esempio precedente la matrice A è quasi separabile superiormente di ordine 3 e inferiormente di ordine 1. In generale si parlerà di matrici con struttura di rango per intendere che le matrici sono quasi separabili.

Osservazione 1.13. Si consideri ora una correzione di rango 1 di una matrice unitaria (come le matrici *companion*)

$$A = H + uv^H \qquad \text{con } H \text{ unitaria}$$

Allora si inizia subito con l'osservare che AA^H è una correzione di rango 2 della matrice identica, infatti

$$\begin{aligned} AA^H &= (H + uv^H)(H^H + vu^H) \\ &= I + (Hvu^H + uv^H H + uv^H vu^H) \qquad \text{quello tra parentesi ha rango 3} \\ &= I + [uv^H H^H + (Hv + u(v^H v))u^H] \quad \text{quello tra parentesi quadre ha rango 2} \end{aligned}$$

Questo servirà più avanti.

Esempio 1.4.1. Si consideri

$$A = H + uv^H \qquad \text{con } H \text{ unitaria}$$

Come sempre si assume che A sia in forma di Hessenberg, allora

$$H = \begin{pmatrix} * & * & * & * & * & * & * \\ * & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & & * & * & * & * & * \\ & & & * & * & * & * \\ & & & & * & * & * \\ & & & & & * & * \end{pmatrix} - uv^H$$

Quindi l'indice di quasi seperabilità inferiore di H è 2. Se H fosse Hermitiana si potrebbe dire che anche l'indice di quasi separabilità superiore è 2, nel caso delle matrice unitarie non è chiaro se questa proprietà continua a valere. Daltronde ci sono dei lavori che mostrano come dalla conoscenza della parte triangolare inferiore di una matrice unitaria si possa ricostruire la parte triangolare superiore, a cosa questo serva sarà chiaro quando si daranno dei teoremi di rappresentazione per le matrici con struttura di rango. E' importante non fraintendere quanto detto, la ricostruzione di una matrice unitaria a partire dalla triangolare inferiore non è unica, basti pensare alla seguente matrice

$$\begin{pmatrix} * & * & * & * & * & * & * \\ 1 & * & * & * & * & * & * \\ 0 & 1 & * & * & * & * & * \\ 0 & 0 & 1 & * & * & * & * \\ 0 & 0 & 0 & 1 & * & * & * \\ 0 & 0 & 0 & 0 & 1 & * & * \\ 0 & 0 & 0 & 0 & 0 & 1 & * \end{pmatrix}$$

Supponiamo di voler determinare gli $*$ che rendono unitaria la matrice, allora basterà prendere un qualsiasi $\alpha \in \mathbb{C}$ di norma unitaria

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \alpha \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

E questa è una matrice unitaria. Quello che è importante è che per la matrici unitarie il grado di quasi separabilità inferiore sia uguale al grado di semi separabilità superiore. Quindi in questo esempio H ha grado di semi separabilità superiore 2.

1.4.2 Inversione di matrici tridiagonali

L'obbiettivo che ci si pone ora è l'inversione delle matrici tridiagonali simmetriche irriducibili.

$$T = \begin{pmatrix} \alpha_1 & \beta_1 & & & \\ \beta_1 & \alpha_2 & \beta_2 & & \\ & \ddots & \ddots & \ddots & \\ & & \beta_{n-2} & \alpha_{n-1} & \beta_{n-1} \\ & & & \beta_{n-1} & \alpha_n \end{pmatrix}$$

Si osserva subito che queste matrici hanno ordine di quasi separabilità 1. Dato che queste matrici sono sparse e in generale si rappresentano con due vettori, in generale ci si aspetta che anche le loro inverse abbiano questa proprietà, daltronde si mostrerà che questo è falso in generale per quanto riguarda la sparsità, ma è possibile comunque memorizzare l'inversa con $O(n)$, ovvero esisterà una rappresentazione (che chiameremo parametrizzazione della matrice) che prevede l'utilizzo di pochi vettori. In generale si parla di matrici *Data Sparse* per indicare matrici che si possono memorizzare in $O(n)$. Si vedrà che questo è collegato fortemente con la struttura di rango introdotta nella sezione precedente. E' però necessario introdurre le rotazioni di Givens per eseguire in modo agevole la fattorizzazione QR.

Definizione 1.4.3. Si definisce rotazione di Givens una matrice che è l'identica tranne che per un blocco diagonale 2×2, quindi

$$\left(\begin{array}{c|cc|c} I_k & & & \\ \hline & c & s & \\ & -s & c & \\ \hline & & & I_{n-k-2} \end{array}\right) \qquad \text{con} \qquad c^2 + s^2 = 1$$

Non è difficile provare che esiste sempre una successione di rotazioni di Givens che trasformano qualsiasi vettore in un multiplo di e_1, proprio come con le matrici di Householder. Come esempio si mostra quello base ma la generalizzazione è immediata

$$\begin{pmatrix} \frac{\bar{x}}{\sqrt{x^2+y^2}} & \frac{\bar{y}}{\sqrt{x^2+y^2}} \\ -\frac{y}{\sqrt{x^2+y^2}} & \frac{x}{\sqrt{x^2+y^2}} \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} \sqrt{x^2+y^2} \\ 0 \end{pmatrix}$$

In realtà le rotazioni di Givens si possono definire in modo molto più generale ma non è necessario in questo contesto.

Si inizia subito con l'osservare che l'inversa della matrice T sarà ancora simmetrica. Se si effettuano in successione tutte le rotazioni di Givens per ottenere la fattorizzazione QR si trova che

$$\left(\begin{array}{c|cc} I_{n-2} & & \\ \hline & c_{n-1} & s_{n-1} \\ & -s_{n-1} & c_{n-1} \end{array}\right) \cdots \left(\begin{array}{c|cc|c} 1 & & & \\ \hline & c_1 & s_1 & \\ & -s_1 & c_1 & \\ \hline & & & I_{n-2} \end{array}\right) \left(\begin{array}{cc|c} c_1 & s_1 & \\ -s_1 & c_1 & \\ \hline & & I_{n-2} \end{array}\right) A =$$

$$\begin{pmatrix} \gamma_1 & \delta_1 & \theta_1 & & & \\ & \gamma_2 & \delta_2 & \theta_2 & & \\ & & \ddots & \ddots & \ddots & \\ & & & \ddots & \ddots & \theta_{n-2} \\ & & & & \ddots & \delta_{n-1} \\ & & & & & \gamma_n \end{pmatrix}$$

Quindi si è ottenuto $Q^H T = R$ da questo si trova la fattorizzazione $T = QR$ allora

$$T^{-1} = R^{-1} Q^H$$

Quindi bisogna capire come è fatta $R^{-1}Q^H$. Chiaramente l'inversa di una triangolare superiore sarà una triangolare superiore, per ora non è necessario sapere altro. Per comodità di notazione nelle matrici di Givens si pone $b_j := -s_j$ e $a_j := c_j$, mentre con r_j si denotano gli elementi diagonale di R^{-1}, allora

$$\begin{pmatrix} * & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & & * & * & * & * & * \\ & & & * & * & * & * \\ & & & & * & * & * \\ & & & & & * & * \\ & & & & & & * \end{pmatrix} \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & 1 & & \\ & & & & & a_{n-1} & -b_{n-1} \\ & & & & & b_{n-1} & a_{n-1} \end{pmatrix} = \begin{pmatrix} * & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & & * & * & * & * & * \\ & & & * & * & * & * \\ & & & & * & * & * \\ & & & & & * & * \\ & & & & & r_n b_{n-1} & * \end{pmatrix}$$

Proseguendo con le rotazioni di Givens

$$\begin{pmatrix} * & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & & * & * & * & * & * \\ & & & * & * & * & * \\ & & & & * & * & * \\ & & & & & * & * \\ & & & & & r_n b_{n-1} & * \end{pmatrix} \begin{pmatrix} 1 & & & & & & \\ & 1 & & & & & \\ & & 1 & & & & \\ & & & 1 & & & \\ & & & & a_{n-2} & -b_{n-2} & \\ & & & & b_{n-2} & a_{n-2} & \\ & & & & & & 1 \end{pmatrix} = \begin{pmatrix} * & * & * & * & * & * & * \\ & * & * & * & * & * & * \\ & & * & * & * & * & * \\ & & & * & * & * & * \\ & & & & * & * & * \\ & & & & r_{n-1} b_{n-2} & * & * \\ & & & & r_n b_n & r_n b_{n-1} & * \end{pmatrix}$$

Si può proseguire in questo modo e trovare che in generale

$$(T^{-1})_{i,j} = r_i b_{i-1} \dots b_j a_{j-1} \qquad \text{per } i > j$$

Dove si definisce $a_0 := 1$ e $b_0 := 1$. E' noto che T^{-1} è simmetrica, quindi è sufficiente conoscere la parte inferiore. Di conseguenza è possibile rappresentare T^{-1} con $O(n)$, ovvero è sufficiente usare tre vettori, ovvero

$$(r_1, \ldots, r_n)$$
$$(a_1, \ldots, a_{n-1})$$
$$(b_1, \ldots, b_{n-1})$$

Questi vettori sono anche detti *parametrizzazione* della matrice T^{-1}. Questo è come anticipato all'inizio un caso di *Sparse Data*, infatti la matrice non è sparsa ma si rappresenta in $O(n)$. Non è difficile mostrare che questa matrice ha una struttura di rango con ordine di quasi separabilità 1 (per vederlo basta scrivere una matrice con questa struttura ed è evidente, una dimostrazione la si potrebbe fare per induzione sulla dimensione). Quello che è importante è che in questo caso la struttura di rango si è conservata con l'inversione.

Definizione 1.4.4. Una matrice è semiseparabile se è l'invesa di una matrice tridiagonale simmetrica irriducibile.

Teorema 1.4.1. Sia $P \in \mathbb{R}^{n\times n}$ tridiagonale, simmetrica e irriducibile, allora esistono $(x_i)_{1\leq i\leq n}$ e $(y_i)_{1\leq i\leq n}$ tale che

$$(T^{-1})_{i,j} = x_i y_j \qquad \text{con } i \geq j$$

Con il teorema appena esposto tutte le osservazioni fatte sin ora diventano banali. Ad ogni modo la rappresentazione a cui fa riferimento l'ultimo teorema non è unica e non è stabile numericamente, quindi si preferisce usare l'approccio mostrato con le rotazioni di Givens.

1.4.3 Algoritmo QR per matrici con struttura di rango

Definizione 1.4.5. Una matrice $F = (f_{i,j})_{i,j=1,\dots,n}$ si dice quasi separabile di ordine k se

$$f_{i,j} = \begin{cases} p_i^T a_{i-1} \dots a_{j+1} q_j & \text{se } 1 \le j < i \le n \\ d_i & \text{se } i = j \\ g_i^T b_{i+1} \dots b_{j-1} h_j & \text{se } 1 \le i < j \le n \end{cases}$$

Dove $g_i, p_i, q_j, h_j \in \mathbb{C}^k$, mentre $a_i, b_i \in \mathbb{C}^{k \times k}$

Teorema 1.4.2. Se F ha ordine di quasi separabilità superiore h allora è quasi separabile di ordine h

Teorema 1.4.3. Se F ha ordine di quasi separabilità inferiore k allora è quasi separabile di ordine k

Esempio 1.4.2. Si consideri ora il caso companion, ovvero si rappresenta come una correzione di rango uno di una matrice unitaria: $A = Q + uv^T$. In questo caso questa struttura (correzione di rango uno di una unitaria) si conserva durante l'algoritmo QR, inoltre si è visto che questa matrice ha ordine di quasi separabilità 3, quindi si rappresenta come

$$A_k = \begin{pmatrix} d_1^{(k)} & \dots & {g_1^{(k)}}^T B_{i+1}^{(k)} \dots B_{j-1}^{(k)} h_j^{(k)} & \dots \\ \beta_1^{(k)} & \ddots & & \vdots \\ & \ddots & & \\ & & & \\ & & & \\ & & & \\ & & & \\ & & \beta_{n-1}^{(k)} & d_n^{(k)} \end{pmatrix}$$

Dove $B_i^{(k)} \in \mathbb{C}^{3\times 3}$, quindi è possibile associare alla matrice A_i la sua parametrizzazione

$$(B_i^{(k)}, d_i^{(k)}, q_i^{(k)}, \beta_i^{(k)}, h_i^{(k)})$$

Quindi il problema sarà capire come si trasforma la parametrizzazione dopo un passo dell'algoritmo QR, ovvero quale sarà la parametrizzazione di A_{k+1}, quindi si vorrà sapere

$$(B_i^{(k)}, d_i^{(k)}, q_i^{(k)}, \beta_i^{(k)}, h_i^{(k)}) \to (B_i^{(k+1)}, d_i^{(k+1)}, q_i^{(k+1)}, \beta_i^{(k+1)}, h_i^{(k+1)})$$

Si osserva ancora che A_i è una matrice piena ma la si rappresenta in memoria con $O(n)$.

Esempio 1.4.3. Si consideri una correzione di rango uno di una matrice Hermitiana $A = H + uv^H$, si è visto che questa struttura si conserva durante l'algoritmo QR, quindi $A_k = H_k + u_k v_k^H$.

$$A_k = \begin{pmatrix} d_1^{(k)} & \gamma_1^{(k)} & \dots & {g_1^{(k)}}^T B_{i+1}^{(k)} \dots B_{j-1}^{(k)} h_j^{(k)} & \dots \\ \beta_1^{(k)} & & \ddots & & \vdots \\ & & \ddots & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \gamma_{n-1}^{(k)} \\ & & & \beta_{n-1}^{(k)} & d_n^{(k)} \end{pmatrix}$$

In questo caso $B_i^{(k)} \in \mathbb{C}^{2\times 2}$.

Anche in questo caso ci si chiede come si aggiorna la parametrizzazione dopo un passo dell'algoritmo QR

$$(B_i^{(k)}, d_i^{(k)}, q_i^{(k)}, \beta_i^{(k)}, h_i^{(k)}, \gamma_i^{(k)}) \to (B_i^{(k+1)}, d_i^{(k+1)}, q_i^{(k+1)}, \beta_i^{(k+1)}, h_i^{(k+1)}, \gamma_i^{(k+1)})$$

In generale sarà considerato l'algoritmo QR con shift

$$\begin{cases} (A_k - \alpha_k I) = Q_k R_k \\ A_{k+1} = R_k Q_k + \alpha_k I \end{cases}$$

Chiaramente si da per scontato che A è già in forma di Hessenberg e per ora si considera in caso companion.

$$A_k - \alpha_k I = \begin{pmatrix} d_1 & g_1^T h_2 & g_1^T B_2 & g_1^T B_2 B_3 h_4 & g_1^T B_2 B_3 B_4 h_5 & \dots \\ \beta_1 & d_2 & g_2^T h_3 & g_2^T B_3 h_4 & g_2^T B_3 B_4 h_5 & \vdots \\ & \beta_2 & d_3 & g_3^T h_4 & g_3^T B_4 h_5 & \\ & & \ddots & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & \beta_{n-1} & d_n \end{pmatrix}$$

Per avere la fattorizzazione QR di questa matrice si dovranno usare le rotazioni di Givens, quindi si dovà moltiplicare per una matrice

$$G_1 = \left(\begin{array}{cc|c} c & -s & \\ s & c & \\ \hline & & I_{n-2} \end{array}\right)$$

Quindi in sostanza si farà una combinazione lineare tra le prime due righe, quindi ci si chiede come viene modificata la struttura di rango. Si definiscono

$$\begin{aligned} \widehat{g_1}^T &:= c g_1^T B_2 - s g_2^T \\ \widehat{g_2}^T &:= s g_1^T B_2 + s g_2^T \end{aligned}$$

Quindi si ha

$$G_1(A_k - \alpha_k I) = \begin{pmatrix} \widehat{d_1} & g_1^T h_2 & \widehat{g_1}^T B_2 & \widehat{g_1}^T B_2 B_3 h_4 & \widehat{g_1}^T B_2 B_3 B_4 h_5 & \dots \\ & \widehat{d_2} & \widehat{g_2}^T h_3 & \widehat{g_2}^T B_3 h_4 & \widehat{g_2}^T B_3 B_4 h_5 & \vdots \\ & \beta_2 & d_3 & g_3^T h_4 & g_3^T B_4 h_5 & \\ & & \ddots & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & \beta_{n-1} & d_n \end{pmatrix}$$

Quindi si procede con le rotazioni di Givens fino ad ottenere (per comodità si fissa $k = 1$)

$$G_{n-1} \dots G_1 (A - \alpha I) = R$$

Pertanto si trova

$$A - \alpha I = G_1^H \dots G_{n-1}^H R$$

Quindi per il passo successivo

$$A_1 = R G_1^H \dots G_{n-1}^H$$

Quindi è sensato chiedersi quale forma abbia A_1, per non complicare troppo le notazioni si utilizzano le stesse lettere usate prima, quindi poniamo

$$R = \begin{pmatrix} d_1 & g_1^T h_2 & g_1^T B_2 & g_1^T B_2 B_3 h_4 & g_1^T B_2 B_3 B_4 h_5 & \dots \\ & d_2 & g_2^T h_3 & g_2^T B_3 h_4 & g_2^T B_3 B_4 h_5 & \vdots \\ & & d_3 & g_3^T h_4 & g_3^T B_4 h_5 & \\ & & \ddots & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & d_n \end{pmatrix}$$

Dove come già detto i vari parametri $h_i, B_i, \dots$ non hanno nulla a che fare con quelli scritti prima. Ad ogni modo per calcolare A_1 bisogna dunque eseguire il prodotto

$$A_1 = R G_1^H \dots G_{n-1}^H$$

Quindi al primo passo bisogna eseguire il prodotto $R G_1^H$, questa operazione fa una combinazione lineare delle prime due colonne (si ripete un ragionamento simile a quello appena fatto per la combinazione delle prime due righe). Quindi compare un elemento nella sottodiagonale. Si procede con le rotazioni di Givens e si ritorna ad una matrice A_1 in forma di Hessenberg. Ci sono problemi sul fatto che la struttura di rango sembra crescere ma in reatà si può osservare che ciò non avviene a meno di effettuare una *compressione* ma si evita di dare dettagli tecnici. Quello che è importate è che il costo di un passo del metodo QR in questo caso ha costo $O(n)$, quindi si riescono a trovare gli autovalori in $O(n^2)$.

1.4.4 GEP strutturati

Consideriamo il GEP (generalized eigenvalue problem), ovvero i problemi della forma

$$Ax = \lambda B$$

Il metodo a cui si farà riferimento sarà il *QZ*. Si assumerà inoltre A di Hessenberg e B triangolare superiore (ci si può sempre ricondurre a questostato), allora si consederano le trasformazioni del tipo

$$(A_0, B_0) \to (A_1.B_1)$$

Dove

$$\begin{cases} A_1 = U_1 A_0 Z_1 \\ B_1 = U_1 B_0 Z_1 \end{cases}$$

Con U_1 e Z_1 unitarie.
Supponiamo ora di lavorare con matrici che sono correzioni di rango basso di matrici unitarie, quindi di avere a che fare con strutture di rango, per semplicità si considerano le correzioni di rango 1, allora

$$A_0 = Q_0 + uv^T$$
$$B_0 = W_0 + zq^T$$

Allora questa struttura si conserva ad ogni passo dell'algoritmo *QZ*. In particolare si è giò osservato che A_i ha come ordine inferiore 1 e come ordine superiore 3 (ci si riferisce alla definizioni 1.4.1 e 1.4.2), Mentre la matrice B_i ha ordine superiore 2. un caso interessante è il *companion pencil*, ovvero

$$A_0 = \begin{pmatrix} & & & -p_0 \\ 1 & & & -p_1 \\ & \ddots & & \vdots \\ & & 1 & -p_{n-1} \end{pmatrix} \quad B_0 = \begin{pmatrix} 1 & & & \\ & \ddots & & \\ & & 1 & \\ & & & p_n \end{pmatrix}$$

Si è già osservato che A_0 è una correzione di rango 1 della matrice circolante e la matrice B_0 è una correzione di rango 1 della matrice identica, quindi siamo nel caso appena descritto.

Si supponga ora di voler trattare il caso Hermitiano, ovvero

$$A_0 = H_0 + uv^T$$
$$B_0 = S_0 + zq^T$$

Appunto con H_0 e S_0 Hermitiane, allora in questo caso si può mostrare che si perdono queste strutture durante l'esecuzione del *QZ*.

Un caso che si sa affrontare è quello in cui

$$\begin{cases} A_0 = H_0 + uv^T \\ B_0 = I + zq^T \end{cases}$$

Allora si trova subito che

$$A_1 = U_1 H_0 Z_1 + U_1 uv^T Z_1$$
$$B_1 = U_1 Z_1 + U_1 zq^T Z_1$$

Ai fini dell'algoritmo QZ si è interessati (supponendo B invertibile) a $A_1B_1^{-1}$, pertanto si vogliono delle informazioni su questa matrice

$$\begin{aligned} B_1 &= U_1(I + zq^T)Z_1 \\ B_1^{-1} &= Z_1^H(I + zq^T)U_1^H \\ &= Z_1^H(I + \sigma zq^T)U_1^H \end{aligned}$$

Dove si è usato il fatto che si sanno calcolare le inverse di matrici elementari. Quindi

$$\begin{aligned} A_1B_1^{-1} &= (U_1H_0Z_1 + U_1uv^TZ_1)Z_1^H(I + \sigma zq^T)U_1^H \\ &= (U_1H_0 + U_1uv^T)(I + \sigma zq^T)U_1^H \\ &= U_1H_0(I + \sigma zq^T)U_1^H + U_1uv^T(I + \sigma zq^T)U_1^H \\ &= U_1H_0U_1^H + \text{qualcosa di rango 1} + \text{qualcosaltro di rango 1} \\ &= U_1H_0U_1^H + \text{qualcosa di rango 2} \end{aligned}$$

Quindi $A_1B_1^{-1}$ è una correzione di rango 2 di una matrice hermitiana, in particolare avrà ordine di quasi separabilità superiore 5 e inferiore 3. Pertanto questa struttura si riesce a conservare ad ogni passo.

1.4.5 PEP strutturati e linearizzazioni

Si considerino ora i PEP (polynomila eigenvalue problem), ovvero trovare le soluzioni di

$$\det\left(\sum_{i=0}^{m} A_i\lambda^i\right) = 0 \qquad A_j \in \mathbb{C}^{k\times k}$$

Per ora ci si limita ad esaminare il caso *QEP* (quadratic eigenvalue problem), ovvero

$$\det(A_0 + \lambda A_1 + \lambda^2 A_2) = 0$$

E' conveniente annunciare subito l'obbiettivo che ci si pone quando si vuole trattare i *PEP*, quello che si cerca di fare è ottenere una *linearizzazione*, ovvero ricondurre un *PEP* ad un *GEP*, mostriamo per ora solo come ottenere una linearizzazione di un *QEP*, quindi si trasformerà il problema

$$\det(A_0 + \lambda A_1 + \lambda^2 A_2) = 0$$

nel problema

$$\det\left[\begin{pmatrix} 0 & -A_0 \\ I_k & -A_1 \end{pmatrix} - \lambda \begin{pmatrix} I_k & 0 \\ 0 & A_2 \end{pmatrix}\right] = 0$$

Si osserva subito che questa non è altro che la generalizzazione della costruzione della matrice companion. Mostriamo ora la relazione che c'è tra il *QEP* e il *GEP* appena scritti. E' possibile estendere la definizione di determinante pensando alla matrice appena scritta come una matrice a coefficienti in un dominio di integrità commutativo e vedere quest'ultima espressione come

$$\det\begin{pmatrix} -\lambda I_k & -A_0 \\ I_k & -A_1 - \lambda A_2 \end{pmatrix} \in \mathbb{F}^{2k\times 2k} \qquad \text{con} \qquad \mathbb{F} = \mathbb{C}[\lambda]$$

Pertanto è possibile eseguire le *mosse di Gauss generalizzate* per calcolare tale determinante, quindi con una matrice di permutazione P che ha quindi determinante ± 1 si ha che

$$P\begin{pmatrix} -\lambda I_k & -A_0 \\ I_k & -A_1 - \lambda A_2 \end{pmatrix} = \begin{pmatrix} I_k & -A_1 + \lambda A_2 \\ \lambda I_k & -A_0 \end{pmatrix}$$

Moltiplico per la matrice di eliminazione gaussiana E che aggiunge λ volte la prima riga all'ultima riga, quindi

$$EP\begin{pmatrix} -\lambda I_k & -A_0 \\ I_k & -A_1-\lambda A_2 \end{pmatrix} = \begin{pmatrix} I_k & -A_1+\lambda A_2 \\ \lambda I_k & -A_0-\lambda A_1-\lambda^2 A_2 \end{pmatrix}$$

Posso ora moltiplicare a destra per la matrice G che esegue l'eliminazione gaussiana sulle colonne

$$G = \begin{pmatrix} I_k & A_1+A_2 \\ 0 & I_k \end{pmatrix}$$

Allora

$$\begin{aligned} EP\begin{pmatrix} -\lambda I_k & -A_0 \\ I_k & -A_1-\lambda A_2 \end{pmatrix} G &= \begin{pmatrix} I_k & 0 \\ 0 & -A_0-\lambda A_1-\lambda^2 A_2 \end{pmatrix} \\ &= \begin{pmatrix} I_k & 0 \\ 0 & -p(\lambda) \end{pmatrix} \end{aligned}$$

Dove chiaramente si intende

$$p(\lambda) = A_0 + \lambda A_1 + \lambda^2 A_2$$

In conclusione

$$\det(p(\lambda)) = \pm\det\left(\begin{pmatrix} 0 & A_0 \\ I_k & A_1 \end{pmatrix} - \lambda\begin{pmatrix} I_k & 0 \\ 0 & A_2 \end{pmatrix}\right)$$

Quindi e' possibile risolvere il problema applicando il QZ al pencil

$$\begin{pmatrix} 0 & A_0 \\ I_k & A_1 \end{pmatrix} - \lambda\begin{pmatrix} I_k & 0 \\ 0 & A_2 \end{pmatrix}$$

Quindi si è ridotto il QEP in un GEP a patto di aver raddoppiato la dimensione del problema. Questo processo è noto come linearizzazione.

Definizione 1.4.6 (Linearizzazione)**.** Dato il PEP

$$\det(p(\lambda)) = 0$$

Diremo che il pencil $E - \lambda F$ è una linearizzazione del PEP se esistono $M, N \in \mathbb{F}^{nk\times nk}$ tali che

$$M(E-\lambda F)N = \begin{pmatrix} I_{n(k-1)} & \\ & p(\lambda) \end{pmatrix}$$

Quella appena mostrata è detta linearizzazione di tipo *companion*.

Esempio 1.4.4 (PEP palindromo)**.** Si consideri il PEP

$$p(\lambda) = A_0 + \lambda A_1 + \lambda^2 A_0$$

Si osserva che questo PEP,ha delle particolari proprietà, infatti

$$\det(p(\lambda)) = \det\left(\lambda^2 p\left(\frac{1}{\lambda}\right)\right) = \lambda^2\det\left(\left(\frac{1}{\lambda}\right)\right)$$

Quindi se λ è un autovalore allora lo è anche $\frac{1}{\lambda}$. In questo caso si avebbe interesse nel trovare linearizzazioni del tipo $A - \lambda A^T$ dato che si ha appunto

$$\det(A-\lambda A^T) = \pm\det\left(\lambda\left(A-\frac{1}{\lambda}A^T\right)\right) = \pm\lambda\det\left(A-\frac{1}{\lambda}A^T\right)$$

Quindi la linearizzazione ha la stessa proprietà del PEP, ovvero si hanno gli autovalori in coppia $\left(\lambda, \frac{1}{\lambda}\right)$.

Osservazione 1.14. Cosa su può dire in generale sulle strutture che si possono conservare? Quali sono i problemi legati a questo aspetto? Attualmente si sa pochissimo. Si consideri $A \in \mathbb{C}^{k\times k}$ con k grande e si consideri $n << k$ e si consideri il *PEP* dato dal polinomio

$$p(\lambda) = \sum_{i=1}^{n} A_i \lambda^i$$

Allora si consideri una linearizzazione di tipo companion

$$\begin{pmatrix} & & & -A_0 \\ I & & & \\ & \ddots & & \vdots \\ & & I & -A_{n-1} \end{pmatrix} + \lambda \begin{pmatrix} I & & & \\ & \ddots & & \\ & & I & \\ & & & A_n \end{pmatrix}$$

Entrambe le matrici del matrix pencil sono correzioni di rango k di matrici unitarie, ma k è grande ed il problema è proprio questo. Quindi per avere vantaggi dal punto di vista strutturale ci si dovrebbe trovare nella situazione opposta, ovvero $n >> k$ e su questo aspetto ci sono alcuni lavori.

Altri problemi più generali che si trovano, più generali del *PEP* sono ad esempio

$$\det(f(\lambda)) = 0$$

Dove f è una qualsiasi funzione piuttosto che un polinomio, ad esempio dalla teoria sulle *DDE* (delay differential equation) si trova il problema

$$\det(A_0 + \lambda A_1 + e^{\lambda} A_2) = 0$$

Si osserva che in questo caso gli autovalori sono un'infinità numerabile.

Oppure potrebbe aver interesse risolvere un problema di tipo razionale

$$\det\left(\frac{A}{\gamma - \lambda} + \frac{A}{\alpha - \lambda} + \frac{A}{\beta - \lambda}\right) = 0$$

Capitolo 2

Sistemi lineari in domini di integrità

2.1 Introduzione

Si considerino due curve planari razionali, ovvero le funzioni

$$\begin{array}{rccccc} \psi_1: & [0,1] & \to & & \mathbb{R}^2 & \\ & t & \to & \left(\frac{p_1(t)}{q_1(t)} & , & \frac{p_2(t)}{q_2(t)}\right) \end{array}$$

Dove $p_1(t), q_1(t), p_2(t), q_2(t) \in \mathbb{R}[x]$ (sono polinomi).

$$\begin{array}{rccccc} \psi_2: & [0,1] & \to & & \mathbb{R}^2 & \\ & s & \to & \left(\frac{g_1(s)}{t_1(s)} & , & \frac{g_2(s)}{t_2(s)}\right) \end{array}$$

Dove $g_1(t), t_1(t), g_2(t), t_2(t) \in \mathbb{R}[x]$. Ci si chiede se queste due curve si intersecano, quindi se esistono s, t tali che

$$\begin{cases} \frac{p_1(t)}{q_1(t)} = \frac{g_1(s)}{t_1(s)} \\ \frac{p_2(t)}{q_2(t)} = \frac{g_2(s)}{t_2(s)} \end{cases}$$

Ovvero

$$\begin{cases} p_1(t)t_1(s) - g_1(s)q_1(t) = 0 \\ p_2(t)t_2(s) - g_2(s)q_2(t) = 0 \end{cases}$$

Quindi posto

$$\begin{array}{rl} f_1(s,t) := & p_1(t)t_1(s) - g_1(s)q_1(t) \\ f_2(s,t) := & p_2(t)t_2(s) - g_2(s)q_2(t) \end{array}$$

Allora $f_1, f_2 \in \mathbb{R}[s,t]$, ci si chiede dunque se questi due polinomi bivariati (ovvero con due variabili) hanno radici in comune, quindi si vorrà risolvere

$$\begin{cases} f_1(s,t) = 0 \\ f_2(s,t) = 0 \end{cases}$$

Quindi bisogna trovare un criterio per trovare le radici in comune di due polinomi bivariati.

2.2 Radici in comune di due polinomi

2.2.1 Matrici risultati

Si inizia con l'affrontare il problema nel caso di polinomi univariati (con una sola variabile), dunque si considerino i polinomi

$$p(x) = p_0 + p_1 x + \cdots + p_n x^n \qquad p_n \neq 0$$
$$q(x) = q_0 + q_1 x + \cdots + q_m x^m \qquad q_m \neq 0$$

Osservazione 2.1. Una prima idea può esser quella di calcolare tutti gli zeri di entrambi i polinomi e di confrontarli, ma ci sono vari problemi computazionali, ad esempio il confronto tra due zeri non è semplice da effettuare dato che, ammesso che ci sia uno zero in comune, avrebbe un'approssimazione differente se calcolato come zero di p o di q. Supponendo pure di poterlo fare, si osserva che questa è una procedura non razionale e quindi si avranno difficoltà a generalizzare queste procedure nel caso di polinomi bivariati.

Si supponga $n \geq m$, si osserva subito che

$$\exists \xi \quad \text{t.c.} \quad p(\xi) = q(\xi) \iff GCD(p(x), q(x)) \neq 1$$

Quindi si potrebbe pensare di utilizzare l'algoritmo di Euclide.

Richiamo 2.2.1 (Algoritmo di Euclide)**.** L'algoritmo di Euclide per la ricerca del GCD (massimoc comun divisore) tra $p(x)$ e $q(x)$ è il seguente

$$r_0(x) = p(x)$$
$$r_1(x) = q(x)$$
$$r_j(x) = s_j(x) r_{j+1} + r_{j+2}(x) \qquad j = 0, 1, \ldots$$

Dove si è effettuata la divisione euclidea ed $r_{j+2}(x)$ è il quoziente e $r_{j+1}(x)$ il resto.

Daltronde dal punto di vista computazionale questo approccio non è conveniente dato che i coefficienti della successione r_j crescono quindi c'è un'amplificazione degli errori. Una soluzione potrebbe essere utilizzare una strategia di pivoting e lavorare in aritmetica esatta ma attualmente non sono note buone strategie di questo genere. Daltronde questo ci mostra che è possibile risolvere il problema in modo razionale, senza quindi dover calcolare gli zeri dei polinomi.

Definizione 2.2.1. Dati due polinomi $p(x), q(x)$, si definisce *matrice di Bezeout* o *bezeoutiano* la matrice $B \in \mathbb{R}^{n \times n}$ i cui coefficienti con i coefficienti del seguente polinomio bivariato

$$\frac{p(x)q(z) - p(z)q(x)}{x - z} = \sum_{i,j=0}^{n-1} b_{i+1,j+1} x^i z^j$$

Quindi appunto

$$B = (b_{i,j})_{i,j=1}^{n}$$

Osservazione 2.2. B è simmetrica (ovvia conseguenza del fatto che i suoi coefficienti sono i coefficienti di un polinomio bivariato simmetrico)

Osservazione 2.3. Vale che

$$\frac{p(x)q(z) - p(z)q(x)}{x - z} = \begin{pmatrix} 1 & x & \ldots & x^{n-1} \end{pmatrix} B \begin{pmatrix} 1 \\ z \\ \vdots \\ z^{n-1} \end{pmatrix}$$

D'ora in avanti, per semplificare i conti, si supporrà che

$$p(x) = \prod_{j=1}^{n}(x - \xi_i) \qquad \text{con} \quad \xi_i \neq \xi_j \ \text{ se } \ i \neq j$$

Ovvero si supporrà che gli zeri siano tutti diversi e reali (questa ipotesi si può abbandonare e con un po più di fatica ottenere tutti i risultati che seguiranno anche senza).

Allora per $h \neq k$ si ha

$$\begin{pmatrix} 1 & \xi_k & \dots & \xi_k^{n-1} \end{pmatrix} B \begin{pmatrix} 1 \\ \xi_h \\ \vdots \\ \xi_h^{n-1} \end{pmatrix} = \frac{p(\xi_h)q(\xi_k) - p(\xi_k)q(\xi_h)}{\xi_k - \xi_h} = 0$$

Quindi B è diagonalizzabile per congruenza, ovvero con la seguente matrice di Vandermonde

$$V := \begin{pmatrix} 1 & \xi_1 & \dots & \xi_1^{n-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \xi_n & \dots & \xi_n^{n-1} \end{pmatrix}$$

si ha

$$VBV^T = \operatorname{diag}(d_1, \dots, d_n)$$

Daltronde ci sono dei problemi per capire quali sono gli elementi diagonali, infatti

$$d_i = \begin{pmatrix} 1 & \xi_i & \dots & \xi_i^{n-1} \end{pmatrix} B \begin{pmatrix} 1 \\ \xi_i \\ \vdots \\ \xi_i^{n-1} \end{pmatrix} = \frac{p(\xi_i)q(\xi_i) - p(\xi_i)q(\xi_i)}{\xi_i - \xi_i} = \frac{0}{0}$$

Quindi per determinarli si userà un passaggio al limite e si sfrutterà la continuità

$$\begin{aligned} d_i &= \lim_{h\to 0} \frac{p(\xi_i + h)q(\xi_i) - p(\xi_i)q(\xi_i + h)}{h} \\ &= \lim_{h\to 0} \left[\frac{p(\xi_i + h) - p(\xi_i)}{h} q(\xi_i) - \frac{q(\xi_i + h) - q(\xi_i)}{h} p(\xi_i) \right] \\ &= p'(\xi_i)q(\xi_i) - q'(\xi_i)p(\xi_i) \\ &= p'(\xi_i)q(\xi_i) \end{aligned}$$

Quindi

$$VBV^T = \begin{pmatrix} p'(\xi_1)q(\xi_1) & & \\ & \ddots & \\ & & p'(\xi_n)q(\xi_n) \end{pmatrix}$$

Pertanto si ha che, essendo in questo caso la matrice di Vandermonde invertibile (zeri distinti)

$$\det B = 0 \iff \exists\, \xi \quad \text{t.c.} \quad p(\xi) = q(\xi) = 0$$

Quindi in conclusione

$$GCD(p(x), q(x)) \neq 1 \iff \det B = 0$$

Quindi si potrebbe utilizzare la fattorizzazione SVD applicata a B trovare il numero di zeri in comune (si deve calcolare la dimensione del $\ker B$), oppure se si vuole un algoritmo robusto si può usare una fattorizzazione LU (con pivoting). Le matrici con questa proprietà sono dette *matrici risultati*. Si mostrerà ora un altro esempio di matrice risultate.

Definizione 2.2.2. Si darà la definizione di matrice di Sylvester con un esempio, la definizione generale sarà subito chiara. Dati due polinomi

$$p(x) = p_0 + p_1 x + p_2 x^2 + p_3 x^3$$
$$q(x) = q_0 + q_1 x + q_2 x^2$$

Allora la matrice di Sylvester sarà

$$S = \begin{pmatrix} p_0 & p_1 & p_2 & p_3 & 0 \\ 0 & p_0 & p_1 & p_2 & p_3 \\ q_0 & q_1 & q_2 & 0 & 0 \\ 0 & q_0 & q_1 & q_2 & 0 \\ 0 & 0 & q_0 & q_1 & q_2 \end{pmatrix}$$

Si osserva subito che se $p(\xi) = q(\xi)$ allora

$$S \begin{pmatrix} \xi \\ \xi^2 \\ \xi^3 \\ \xi^4 \\ \xi^5 \end{pmatrix} = \begin{pmatrix} p(\xi) \\ \xi p(\xi) \\ q(\xi) \\ \xi q(\xi) \\ \xi^2 q(\xi) \end{pmatrix}$$

Si può mostrare che la matrice di Sylvester è una *matrice risultate*, ovvero

$$\det S = 0 \iff GCD(p(x), q(x)) \neq 1$$

Osservazione 2.4. Lo svantaggio di questo approccio è che la dimensione della matrice di Sylvester cresce, la matrice ha dimensione pari alla somma dei gradi dei polinomi, mentre nel caso della matrice di Bezeout la dimensione è quella del maggiore tra i due gradi dei polinomi.

Si mostra ora un'ultimo esempio di matrice risultate, sia $F(p)$ la matrice companion associata al polinomio p, si consideri

$$q(F(p)) = q_0 I + q_1 F(p) + q_2 F(p)^2$$

Gli autovalori di questa matrice sono gli $q(\xi_i)$, quindi questa matrice è singolare se e soltanto se p e q hanno zeri in comune.

Tornando al problema presentato all'inzio, se $f_1, f_2 \in \mathbb{R}[s,t]$ si vuole risolvere

$$\begin{cases} f_1(s,t) = 0 \\ f_2(s,t) = 0 \end{cases}$$

C'è un isomorfismo

$$\begin{aligned} \mathbb{R}[s,t] &\rightarrow \mathbb{F}[t] \\ f(s,t) &\rightarrow g(t) = \sum_{i=0}^{k} g_i(s)t^i \end{aligned}$$

Dove $\mathbb{F} = \mathbb{R}[s]$, allora è possibile associare ad $f_1(s,t) \in \mathbb{R}[s,t]$ il polinomio $g_1 \in \mathbb{F}[t]$ e ad $f_2 \in \mathbb{R}[s,t]$ il polinomio $g_2 \in \mathbb{F}[t]$, quindi è possibile calcolare la matrice di Bezeout di g_1 e g_2 che sarà $\mathbb{F}^{n\times n}$ e per quanto visto prima questa sarà simmetrica. In pratica quello che si sta facendo è vedere polinomi bivariati come polinomi univariati a coefficienti in un anello (quello dei polinomi nell'altra variabile).

A questo punto il modo di procedere è

- Si calcolano
$$\begin{aligned} f_1(s,t) &\rightarrow g_1(t) = \sum_{i=0}^{k} g_i^{(1)}(s)t^i \\ f_2(s,t) &\rightarrow g_1(t) = \sum_{i=0}^{k} g_i^{(2)}(s)t^i \end{aligned}$$
- Si calcolano gli s tali che $\det B = h(s) = 0$
- Ottengono dunque $s_1, \ldots, s_m$ radici di $h(s)$
- Si selezionano le $s_j \in [0,1]$ con $j = 1, \ldots, k$ (se non esistono vuol dire che non ci sono zeri comuni)
- Si calcola $\bar{t}$ tale che
$$\sum_{i=0}^{k} g_i^{(1)}(s_j)\bar{t}^i = \sum_{i=0}^{k} g_i^{(2)}(s_j)\bar{t}^i = 0$$
- Se $\bar{t} \in [0,1]$ allora $\bar{t}, s_j$ è una radice in comune di f_1 e f_2.

Per calcolare $\det B$ bisogna utilizzare l'eliminazione gaussiana in $\mathbb{F}$, daltronde bisogna fare attenzione dato che non si è su un campo e non tutto è invertibile, ad esempio non va bene procedere in modo standard come segue

$$\begin{pmatrix} f & g \\ h & t \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 \\ \frac{h}{g} & 1 \end{pmatrix} \begin{pmatrix} f & g \\ h & t \end{pmatrix} = \begin{pmatrix} f & g \\ 0 & t - \frac{h}{f}g \end{pmatrix}$$

Infatti non è chiaro cosa succede nei punti in cui f si annulla, allora bisogna procedere senza fare divisioni (*varianti division free dell'eliminazione gaussiana*), ovvero

$$\begin{pmatrix} f & g \\ h & t \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 \\ -h & f \end{pmatrix} \begin{pmatrix} f & g \\ h & t \end{pmatrix} = \begin{pmatrix} f & g \\ 0 & -hg + ft \end{pmatrix}$$

2.3 Eliminazione gaussiana su domini di integrità

Si è visto nella sezione precedente che a volte è necessario calcolare il determinante di matrici con elementi in domini di integrità e per farlo si utilizza l'eliminazione gaussiana. Daltronde dato che non tutti gli elementi sono invertibili è necessario eseguire questa operazione senza effettuare divisioni (*division free*). Per semplicità supponiamo che il dominio di integrità sia $\mathbb{Z}$ (si può generalizzare al caso $\mathbb{F}$ qualsiasi, ad esempio $\mathbb{F} = \mathbb{R}[x]$).

Osservazione 2.5. Supponiamo di voler eseguire un'eliminazione gaussiana *division free* "ingenua" su una matrice $A \in \mathbb{Z}^{2\times 2}$, allora

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \to \begin{pmatrix} 1 & 0 \\ -c & a \end{pmatrix} \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} a & b \\ 0 & ad - bc \end{pmatrix}$$

Quello che si osserva è che dato che si lavora su $\mathbb{Z}$ il prodotto tra numeri non è costante ma dipende dalla lunghezza dei numeri (come su $\mathbb{R}[x]$ dipende dal grado dei polinomi). Quindi per avere un costo computazionale non proibitivo si vuole che che i numeri non diventino troppo grandi.

Supponiamo che la matrice $A\mathbb{Z}^{n\times n}$ di cui si vuole eseguire l'eliminazione gaussiana sia tale che la lunghezza dei suoi coefficienti sia q, ovvero $a_{i,j} \in [-2^q, 2^q]$. Definiamo $a_{i,j}^{(s)}$ il coefficiente della matrice A dopo che abbiamo eseguito s passi dell'eliminazione gaussiana nella variante appena definita. Allora

$$\begin{aligned} a_{i,j}^{(0)} &\in [-2^q, 2^q] \\ a_{i,j}^{(1)} &\in [-2 \cdot 2^q, 2 \cdot 2^q] \\ &\vdots \\ a_{i,j}^{(s)} &\in [-2^{2^s \cdot q}, 2^{2^s \cdot q}] \end{aligned}$$

Quindi alla fine dell'eliminazione gaussiana

$$a_{i,j}^{(n)} \in [-2^{2^n \cdot q}, 2^{2^n \cdot q}]$$

Quindi i coefficienti crescono in modo doppiamente esponenziale e quindi la loro lunghezza durante l'algoritmo cresce in modo esponenziale e questo rende assolutamente inaccettabile questo primo approccio.

2.3.1 Variante di Bareis

Si ricorda l'eliminazione gaussiana standard

1	for $k = 1 : n-1$
2	for $i = k : n-1$
3	for $j = k+1 : n$
4	$a_{i,j}^{(k)} = a_{i,j}^{(k-1)} - \frac{a_{i,k}^{(k-1)}}{a_{k,k}^{(k-1)}} a_{k,j}^{(k-1)}$
5	end
6	end
7	end

Quindi in sostanza si ha

$$a_{i,j}^{(k)} = \frac{a_{i,j}^{(k-1)} a_{k,k}^{(k-1)} - a_{i,k}^{(k-1)} a_{k,j}^{(k-1)}}{a_{k,j}^{(k-1)}}$$

Nella versione ingenua dell'eliminazione gaussiana *division free* semplicemente si eliminava la divisione, ovvero

$$a_{i,j}^{(k)} = a_{i,j}^{(k-1)} a_{k,k}^{(k-1)} - a_{i,k}^{(k-1)} a_{k,j}^{(k-1)}$$

Questo approccio lo si è abbandonato appunto per la crescita doppiamente esponenziale dei coefficienti e di conseguenza si ha la crescita esponenziale della lunghezza dei coefficienti.

Si mostrerà ora la variante di Bareis, si inizia con delle osservazioni e richiami sull'eliminazione gaussiana standard. Innanzitutto si fissa la notazione: con $A^{(k)}$ si intende la matrice A dopo che sono stati effettuati k passi dell'eliminazione gaussiana, in particolare $A^{(k)}(i : n, i) = 0$ per $i < k$.
Si osserva che il *pivot* al passo k-esimo non è altro che il rapporto tra i determinanti

$$a_{k,k}^{(k-1)} = \frac{\det A_k}{\det A_{k-1}}$$

Dove A_k e A_{k-1} sono le sottomatrici di testa della matrice $A^{(k)}$.
Inoltre durante l'esecuzione dell'eliminazione gaussiana si ha la successione

$$A = A^{(0)} \to A^{(1)} \to A^{(2)} \to \cdots \to A^{(n-1)}$$

Dove si ha che

$$A^{(k)} = (I - \tau_k e_k^T) A^{(k-1)}$$

conseguenza

$$\tau_k = \left(0, \ldots, 0, \frac{a_{k+1,k}^{(k-1)}}{a_{k,k}^{(k-1)}}, \frac{a_{k+2,k}^{(k-1)}}{a_{k,k}^{(k-1)}}, \ldots, \frac{a_{n,k}^{(k-1)}}{a_{k,k}^{(k-1)}}\right)^T$$

Quindi si ha

$$\begin{aligned} A^{(k)} &= (I - \tau_k e_k^T)A^{(k-1)} \\ &= (I - \tau_k e_k^T)\dots(I - \tau_1 e_1^T)A \\ &= E_k A \end{aligned}$$

Dove si è posto

$$E_k = (I - \tau_k e_k^T)\dots(I - \tau_1 e_1^T)$$

Si osserva dunque che

$$E_k = (I - \tau_k e_k^T)E_{k-1}$$

In generale si avrà dunque

$$E_k = \begin{array}{c} \\ k \end{array} \overset{\overset{k}{\downarrow}}{\begin{pmatrix} 1 & & & & & & & & & \\ * & 1 & & & & & & & & \\ * & * & 1 & & & & & & & \\ * & * & * & 1 & & & & & & \\ * & * & * & * & 1 & & & & & \\ * & * & * & * & * & 1 & & & & \\ * & * & * & * & * & & 1 & & & \\ * & * & * & * & * & & & 1 & & \\ * & * & * & * & * & & & & 1 & \\ * & * & * & * & * & & & & & 1 \end{pmatrix}}$$

Mentre

$$A^{(k)} = \begin{array}{c} \\ k+1 \end{array} \overset{\overset{k+1}{\downarrow}}{\begin{pmatrix} * & * & * & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * & * & * \\ & & * & * & * & * & * & * & * & * \\ & & & * & * & * & * & * & * & * \\ & & & & * & * & * & * & * & * \\ & & & & & * & * & * & * & * \\ & & & & & * & * & * & * & * \\ & & & & & * & * & * & * & * \\ & & & & & * & * & * & * & * \\ & & & & & * & * & * & * & \end{pmatrix}}$$

Consideriamo ora i vettori riga di E_k, in particolare definiamo

$$e_i^{(k)T} = E_k(i+1, 1:i)$$

Allora vale la relazione

$$[e_i^{(k)T}, 0, \dots, 0, 1, 0, \dots, 0]A = A^{(k)}(i, :) \qquad \text{l'1 è nella } k\text{-esisma posizione}$$

Questa relazione la si può riscrivere come

$$e_i^{(k)T} A_{\min\{i,k\}} + [a_{i+1,1}, \dots, a_{i+1,\min\{i,k\}}] = 0$$

Che ancora si può riscrivere come

$$([e_i^{(k)}, 0, \dots, 0]^T + e_{i+1}^T)A = e_i^{(k)T} A(1 : \min\{i : k\}, 1 : n) + [a_{i+1,1}, \dots, a_{i+1,n}]$$

Quindi in questo caso interessa

$$e_i^{(k)T} A_{\min\{i,k\}} + [a_{i+1,1}, \dots, a_{i+1,\min\{i,k\}}] = 0$$

Quindi piuttosto che fare il calcolo come nel processo classico, è possibile risolvere il sistema

$$e_i^{(k)T} A_{\min\{i,k\}} = -[a_{i+1,1}, \dots, a_{i+1,\min\{i,k\}}]$$

Questo è un sistema in $\mathbb{Z}$ quindi la soluzione sarà in $\mathbb{Q}$, se si vuole restare su $\mathbb{Z}$ bisogna sostituire le righe $e_i^{(k)T}$ con delle nuove righe $\widehat{e_i^{(k)T}}$ che soddisfino i corrispondenti sistemi di Cramer, cioè

$$\widehat{e_i^{(k)T}} A_{\min\{i,k\}} = -\det(A_{\min\{i,k\}})[a_{i+1,1}, \dots, a_{i+1,\min\{i,k\}}]$$

Quindi ora si avrà una soluzione in $\mathbb{Z}$, Pertanto la relazione tra le E_k e le $\widehat{E_k}$ è la seguente

$$\widehat{E_k} = D_k E_k$$

Dove

$$D_k = [1, \det A_1, \dots, \det A_{k-1}, \det A_k, \dots, \det A_k]$$

Quindi si avrà una successione

$$A = \widehat{A^{(0)}} \to \widehat{A^{(1)}} \to \cdots \to \widehat{A^{(k-1)}}$$

Allora si trova che

$$\begin{aligned} A^{(k)} &= E_k A \\ \widehat{A^{(k)}} &= \widehat{E_k} A \\ &= D_k A^{(k)} \end{aligned}$$

E' inoltre possibile riscrivere la matrice $\widehat{E_k}$ come

$$\begin{aligned} \widehat{E_k} &= D_k E_k \\ &= D_k (I - \tau_k e_k^T) E_{k-1} \\ &= D_k (I - \tau_k e_k^T) D_{k-1}^{-1} D_{k-1} E_{k-1} \\ &= [D_k (I - \tau_k e_k^T) D_{k-1}^{-1}] \widehat{E_{k-1}} \\ &= M_k \widehat{E_{k-1}} \end{aligned}$$

Dove si è posto

$$\begin{aligned} M_k &= D_k (I - \tau_k e_k^T) D_{k-1}^{-1} \\ &= \begin{pmatrix} I_k & 0 \\ 0 & \frac{\det A_k}{\det A_{k-1}} I_{n-k} \end{pmatrix} - \frac{\det A_k}{\det A_{k-1}} \tau_k e_k^T \end{aligned}$$

Quindi M_k è una sorta di matrice di eliminazione in questa versione generalizzata di eliminazione

gaussiana, si vuol capire come è fatta. Allora

$$\widehat{A^{(k)}} = D_k A^{(k)}$$

$$= \begin{array}{c} \\ \\ \\ \\ \\ \\ k+1 \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \overset{\qquad\qquad\qquad\qquad\qquad\qquad k+1}{} \\ \begin{pmatrix} * & * & * & * & * & * & * & * & * & * \\ & * & * & * & * & * & * & * & * & * \\ & & * & * & * & * & * & * & * & * \\ & & & * & * & * & * & * & * & * \\ & & & & * & * & * & * & * & * \\ & & & & & * & * & * & * & * \\ & & & & & * & * & * & * & * \\ & & & & & * & * & * & * & * \\ & & & & & * & * & * & * & * \\ & & & & & * & * & * & * & \end{pmatrix} \end{array}$$

Si ricorda che

$$\tau_k = \left(0, \ldots, 0, \frac{a_{k+1,k}^{(k-1)}}{a_{k,k}^{(k-1)}}, \frac{a_{k+2,k}^{(k-1)}}{a_{k,k}^{(k-1)}}, \ldots, \frac{a_{n,k}^{(k-1)}}{a_{k,k}^{(k-1)}}\right)^T$$

$$\widehat{A^{(k-1)}} = D_{k-1} A^{(k-1)}$$

$$\widehat{a_{j,k}} = \det A_{k-1} a_{j,k}^{(k-1)}$$

Allora si ha

$$\det A_{k-1} M_k = \begin{pmatrix} \det A_{k-1} I_k & 0 \\ 0 & \det A_k I_{n-k} \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ 0 \\ a_{k+1,k}^{(k-1)} \\ \vdots \\ a_{n,k}^{(k-1)} \end{pmatrix} e_k^T$$

Quindi ora bisogna capire quanto valgono $\det A_{k-1}$ e $\det A_k$, si ricorda che

$$\widehat{A^{(k)}} = D_k A^{(k)}$$

$$D_k = [1, \det A_1, \ldots, \det A_{k-1}, \det A_k, \ldots, \det A_k]$$

Quindi

$$\widehat{a_{k+1,k+1}^{(k)}} = \det A_{k+1}$$

Pertanto

$$a_{k-1,k-1}^{(k-2)} M_k = \begin{pmatrix} a_{k-1,k-1}^{(k-2)} I_k & 0 \\ 0 & a_{k,k}^{(k-1)} I_{n-k} \end{pmatrix} - \begin{pmatrix} 0 \\ \vdots \\ 0 \\ \widehat{a_{k+1,k}^{(k-1)}} \\ \vdots \\ \widehat{a_{n,k}^{(k-1)}} \end{pmatrix} e_k^T$$

Pertanto si ha che ora è possibile esprimere un passo

$$\widehat{A^{(k-1)}} \to \widehat{A^{(k)}}$$

Dove

$$\widehat{a_{i,j}^{(k)}} = \frac{\widehat{a_{k,k}^{(k-1)}}\,\widehat{a_{i,j}^{(k-1)}} - \widehat{a_{i,k}^{(k-1)}}\,\widehat{a_{k,j}^{(k-1)}}}{\widehat{a_{k-1,k-1}^{(k-2)}}} \qquad n-k \leq i,j \leq n$$

Questa è la trasformazione di Bareis, si vede subito che c'è una divisione ma questa divisione è esatta (quindi si resta su $\mathbb{Z}$).

A questo punto è sensato chiedersi se è effettivamente conveniente utilizzare la variante di Bareis, ovvero se così facendo si risolve il problema della crescita esponenziale dei coefficienti, si ricorda che

$$\widehat{E_k} A = \widehat{A^{(k)}}$$

Dove gli elementi E_k sono le soluzioni del sistema lineare

$$\widehat{e_i^{(k)T}} A_{\min\{i,k\}} = -\det(A_{\min\{i,k\}})[a_{i+1,1}, \ldots, a_{i+1,\min\{i,k\}}]$$

Per provare che la crescita dei coefficienti non è esponenziale è necessario utilizzare la seguente

Proposizione 2.3.1 (Disuguaglianza di Hadamard)**.** Data $A \in \mathbb{Z}^{n\times n}$ con $|a_{i,j}| \leq B$ allora vale

$$|\det A| \leq B^n n^{n/2}$$

Quindi nel caso della variante di Bareis si ha

$$\begin{aligned} |a_{i,j}^{(k)}| &\leq (2^q)^n n^{n/2} \\ &= 2^{nq} n^{n/2} \end{aligned}$$

Quindi per la lunghezza $l_{i,j}$ di $a_{i,j}$ si ha (passando al logaritmo)

$$l_{i,j} \leq nq + n\log n$$

Quindi effettivamente non si ha più una crescita esponenziale ma polinomiale.

Osservazione 2.6. Se invece che avere coefficienti in $\mathbb{Z}$ si ha $A \in (\mathbb{R}[x])^{n\times n}$ allora si riesce a controllare il grado dei polinomi (che è equivalente alla lunghezza dei coefficienti quando si è su $\mathbb{Z}$). Il problema in questo caso è che a priori non si riesce a controllare la crescita dei coefficienti dei polinomi (bisogna lavorarci di più, alcune varianti per questo caso ci sono).

Osservazione 2.7. Si ha che

$$\det A_n = a_{n-1,n-1}^{(n-2)}$$

Quindi è ora possibile calcolare il determinate (a meno di prodotto per scalare) di una matrice a coefficienti polinomiali.

La lezione del 17/04/2012 è sostanzialmente la descrizione dei seguenti articoli

- Dinesh Manocha, James Demmel: Algorithms for intersecting parametric and algebraic curves I: simple intersections, ACM Transactions on Graphics Volume 13 Issue 1, Jan. 1994
- D.A. Bini, Ana Marco: Computing curve intersection by means of simultaneous iterations, Numerical Algorithms: Volume 43, Number 2 (2006)

2.4 Smith normal form

Si consideri una matrice a coefficienti in un dominio di integrità $A \in \mathbb{F}^{n\times n}$ con $\mathbb{F} = \mathbb{R}[x]$ oppure $\mathbb{F} = \mathbb{C}[x]$. Si assuma di poter eseguire le seguenti trasformazioni

(I) E' possibile moltiplicare (a sinistra o a destra) per una matrice diagonale con elementi scalari, quindi della forma

$$\mathrm{diag}(1,,\ldots,1,\alpha,1,\ldots,1)$$

(II) E' possibile moltiplicare (a sinistra o a destra) per matrici di permutazione P, quindi è possibile scambiare righe e colonne

(III) E' possibile moltiplicare (a sinistra o a destra) per

$$E = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccc} & & & & j & & & & \end{array} \\ \begin{pmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & 1 & & & & & & \\ & & & 1 & & f(x) & & & \\ & & & & 1 & & & & \\ & & & & & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & 1 & \\ & & & & & & & & 1 \end{pmatrix} \end{array} \begin{array}{c} \\ \\ \\ \\ i \\ \\ \\ \\ \\ \end{array}$$

E' facile mostrare che l'inversa di questa matrice è

$$E = \begin{array}{c} \\ \\ \\ \\ \\ \\ \\ \\ \\ \end{array} \begin{array}{c} \begin{array}{ccccccccc} & & & & j & & & & \end{array} \\ \begin{pmatrix} 1 & & & & & & & & \\ & 1 & & & & & & & \\ & & 1 & & & & & & \\ & & & 1 & & -f(x) & & & \\ & & & & 1 & & & & \\ & & & & & 1 & & & \\ & & & & & & 1 & & \\ & & & & & & & 1 & \\ & & & & & & & & 1 \end{pmatrix} \end{array} \begin{array}{c} \\ \\ \\ \\ i \\ \\ \\ \\ \\ \end{array}$$

Osservazione 2.8. Tutte queste trasformazioni sono determinate da matrici che hanno determinante scalare, ad esempio se $\mathbb{F} = \mathbb{C}[x]$ il determinate è in $\mathbb{C}$.

Teorema 2.4.1 (Smith normal form)**.** Data $A \in \mathbb{F}^{n\times n}$, esistono $P_1, dots, P_r, Q_1, \ldots, Q_l$ matrici di tipo

I,II,III tali che

$$P_1 \dots P_r A Q_1 \dots Q_l = \begin{pmatrix} f_1 & & & & \\ & f_2 & & & \\ & & \ddots & & \\ & & & f_s & \\ & & & & \\ & & & & \\ & & & & \end{pmatrix}$$

Dove (posto che per $a|b$ si intende a divide b) si ha che

$$f_1|f_2|\dots|f_{s-1}|f_s$$

Si definisce

$$\begin{aligned} P &:= P_1 \dots P_r \\ Q &:= Q_1 \dots Q_l \\ B &:= PAQ \end{aligned}$$

Le matrici P e Q hanno determinante costante (cioè scalare, non è un polinomio) e sono dette *matrici unimodulari*. Queste non sono univocamente determinate nella *Smith normal form* ma la matrice B è enivocamente determinata (a meno di multipli scalari).

Dimostrazione. Si mostrerà la tesi per induzione sull'ordine della matrice che si vuole mettere in forma di Smith.

Se $n = 1$ allora non c'è nulla da dimostrare.

Si supponga che la tesi sia vera fino ad $n - 1$, allora si consideri

$$A = \begin{pmatrix} a_{1,1} & \dots & a_{1,n} \\ \vdots & \ddots & \vdots \\ a_{n,1} & \dots & a_{n,n} \end{pmatrix} \in \mathbb{F}^{n\times n} \qquad \text{con} \quad \mathbb{F} = \mathbb{R}[x]$$

Se $A = 0$ allora non c'è nulla da provare, se non si è in questa situazione allora si cerca un elemento $a_{i,j}$ di grado minimo e tramite due trasformazioni di tipo II (permutazioni di righe e colonne) lo si sposta nella posizione $(1, 1)$, quinsi dopo questo passo $a_{1,1}$ è di grado minimo. A questo punto ci sono due casi

(1) $a_{1,1}$ divide tutti gli elementi della matrice, ovvero

$$a_{1,1}|a_{i,j} \qquad \forall i, j = 1, \dots, n$$

Allora $a_{2,1} = m_{2,1}a_{1,1}$, quindi $a_{2,1} - m_{2,1}a_{1,1} = 0$, quindi più in generale tramite trasformazioni di tipo III

$$A \to \begin{pmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & A_1 & \\ 0 & & & \end{pmatrix}$$

Ma allora $a_{1,1}$ continua a dividere tutti gli elementi di A_1, quindi è possibile usare l'ipotesi induttiva.

(2) $a_{1,1}$ non divide tutti gli elementi della matrice, si mostrerà ora che è possibile ricondursi al caso (2). Ci sono due sottocasi

(2a) Esiste un elemento b tale che b non divide a, tramite permutazioni di righe e colonne si porta tale elemento nella posizione $(2,1)$, quindi con l'algoritmo di divisione Euclidea

$$a_{2,1} = qa_{1,1} + r$$

Dove il grado di r è più piccolo del grado di $a_{1,1}$. Con trasformazioni del tipo III è possibile sottrarre alla seconda riga q volte la prima in modo che nella posizione (2,1) compaia r. Pertanto $a_{1,1}$ non sarà più l'elemento di grado minimo, pertanto con permutazione di righe e colonne (trasformazioni del II tipo) è possibile portare l'elemento di grado minore nella posizione $(1,1)$ e si ritorna all'inizio dell'algoritmo.

(2b) Se invece

$$a_{1,1} | a_{i,j} \qquad \text{se} \quad i = 1 \ \text{ oppure } \ j = 1$$

allora con le solite trasformazioni si ha

$$A \to \widehat{A} = \begin{pmatrix} a_{1,1} & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & A_1 & \\ 0 & & & \end{pmatrix}$$

Allora $a_{1,1}$ non può dividere tutti gli elementi di A_1, altrimenti divide tutti gli elementi di A.

Se succede che

$$\deg(a_{1,1}) > \deg\left(a_{i,j}^{(1)}\right) \qquad \text{con} \quad a_{i,j}^{(1)} \in A_1$$

Allora si ricomincia da capo spostando l'elemento di grado minimo nella posizione $(1,1)$.

Se invece

$$\exists b \in A_1 \quad \text{t.c.} \ \ a_{1.1} \quad \text{non divide} \quad b$$

In tal caso con la divisione euclidea

$$b = qa_{1,1} + r \qquad \deg r < \deg a_{1,1}$$

A meno di permutazioni $b = a_{2,2}$, quindi con le solite trasformazioni è possibile far comparire $qa_{1,1}$ nella posizione $(2,1)$ e quindi nella posizione $(2,2)$ invece sarà comparso r e quindi si ricomincia da capo spostando l'elemento di grado più piccolo nella posizione $(1,1)$ e si ricomincia l'algoritmo da capo.

L'algoritmo è finito dato che ad ogni passo almeno un coefficiente della matrice cala di grado, quindi necessariamente non si può andare sotto il grado zero dato che gli unici elementi di grado zero sono gli scalari che dividono tutti i polinomi. □

Osservazione 2.9. Computazionalmente non è molto utile fa forma di Smith dato che è molto costosa e nel fare le divisioni euclidee si possono avere problemi di crescita degle esponenti, questa è utile da un punto di vista teorico.

Ci sono lavori che indagano sulle proprietà struttutali della forma di Smith, ad esempio cosa succede se i polinomi che compaiono nella matrice sono simmetrici, o proprietà simili, ma tralasciamo questi aspetti.