Università degli Studi di Pisa

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Laurea in Matematica

Tesi di Laurea
2005

Fattorizzazione canonica e trasformata discreta di Fourier in un problema di teoria delle code

Candidato
Federico G. Poloni

Relatore
prof. Dario A. Bini
Università di Pisa

Controrelatore
prof. Luca Gemignani
Università di Pisa

Anno Accademico 2004–2005

Capitolo 1

Introduzione al problema

1.1 Il modello di coda

Il problema che tratteremo in questa tesi nasce dalla modellizzazione del concetto intuitivo di coda: ad esempio, il comportamento delle auto in attesa a un casello autostradale, delle persone in fila a uno sportello o, per un esempio più legato all'informatica, dei dati scambiati da un singolo computer con internet.

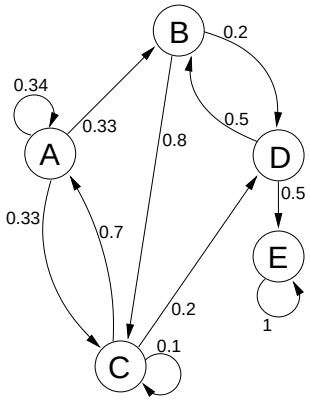
In generale, possiamo chiamare coda un qualunque processo caratterizzato da arrivi casuali nel tempo (pur con una certa legge prestabilita) e da partenze, cioè "persone" che vengono servite e abbandonano la coda. Tra l'arrivo e la partenza c'è un certo periodo di attesa nella coda: ci proponiamo di studiare questo periodo di attesa, rispondendo a domande come *quanto tempo in media dovrò aspettare in coda?* o *qual è la probabilità che ci siano più di n persone in coda al mio arrivo?*

Sono diverse le applicazioni in cui emergono naturalmente delle code: oltre a quelle già citate (auto o persone in attesa di essere servite, pacchetti di dati in arrivo su un *server*), ricordiamo ad esempio le comunicazioni telefoniche via centralino, da cui ebbe origine nel 1909 la teoria delle code con un articolo di A. K. Erlang.

Nella trattazione cercheremo di ridurre al minimo l'utilizzo della teoria della probabilità e la relativa formalizzazione, rimandando il lettore a [7] per ulteriori approfondimenti. Tuttavia, sarà necessario introdurre nella prossima sezione almeno lo strumento principale di cui ci serviremo per trattare il problema, ossia le catene di Markov.

1.2 Catene di Markov

Supponiamo di avere un insieme (finito o numerabile) di *stati* S e delle probabilità di transizione fissate $(P_{ij})_{i,j\in S}$ da uno stato all'altro. Una *catena di Markov* è una successione di variabili aleatorie a valori negli stati, cioè una



$$\begin{bmatrix} 0.34 & 0.33 & 0.33 & 0 & 0 \\ 0 & 0 & 0.8 & 0.2 & 0 \\ 0.7 & 0 & 0.1 & 0.2 & 0 \\ 0 & 0.5 & 0 & 0 & 0.5 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Figura 1.1: esempio di catena di Markov: rappresentazione come grafo e come matrice di transizione

successione di stati $\varphi^{(k)}$ i cui elementi variano in modo aleatorio seguendo le probabilità di transizione che abbiamo fissato. In particolare, una catena di Markov è caratterizzata dalla

Proprietà di Markov. *Lo stato $\varphi^{(k+1)}$ della catena al tempo $k+1$ dipende unicamente dallo stato al tempo k, $\varphi^{(k)}$, (attraverso le probabilità di transizione fissate) e non da tutto il percorso precedente.*

Se chiamiamo $x_i^{(k)}$ la probabilità di essere al tempo k nello stato i e P_{ij} la probabilità di transizione dallo stato i allo stato j, decomponendo sui possibili stati di "partenza" abbiamo la relazione

$$x_j^{(k+1)} = \sum_{i \in S} x_i P_{ij}. \tag{1.1}$$

Questa relazione assume un aspetto molto più compatto se la leggiamo come prodotto di matrici: se $x^{(k)}$ è il vettore riga che ha come componenti gli $x_i^{(k)}$ e P la matrice con elementi P_{ij}, la (1.1) diventa semplicemente

$$x^{(k+1)} = x^{(k)} P. \tag{1.2}$$

La matrice P in questo caso ha elementi compresi nell'intervallo $0 \leq P_{ij} \leq 1$; inoltre è *stocastica*, cioè la somma degli elementi lungo una qualunque riga è uguale a 1:

$$\sum_{j \in S} P_{ij} = 1 \quad \forall i \in S.$$

Questa proprietà, riletta in termini probabilistici, significa semplicemente che partendo da un qualunque stato andiamo sicuramente in un altro stato, quindi la somma delle probabilità di transizione da un i fissato a tutti i j deve necessariamente fare 1.

Notiamo inoltre che anche tutti gli $x^{(k)}$ sono tali che $0 \leq x_i^{(k)} \leq 1$ e $\sum_{i \in S} x_i = 1$. I vettori riga che soddisfano queste due proprietà sono detti *stocastici* in quanto rappresentano una distribuzione di probabilità sull'insieme S.

1.3 Alcuni modelli di code

Supponiamo che il processo della coda si svolga su intervalli di tempo discreti, e che in ogni intervallo temporale k venga servita una persona e arrivino nella coda α_k persone. Il caso più semplice è quello in cui gli arrivi sono indipendenti e con la stessa legge: cioè,

$$\mathbb{P}[\alpha_k = i] = p_i \qquad \text{indipendentemente da } k$$

per una successione p_i data[1]. In questo caso, è naturale modellizzare la coda utilizzando una catena di Markov che ha come stati il numero di persone in coda: $S = \mathbb{N}$ e le probabilità di transizione sono

$$P_{ij} = \begin{cases} p_{j-i+1} & \text{se } i + j > 0 \text{ e } j \geq i - 1, \\ p_0 + p_1 & \text{se } i = j = 0, \\ 0 & \text{se } j < i - 1. \end{cases}$$

Lo zero nell'ultima riga si interpreta dicendo che, visto che viene servita una persona alla volta, il numero di persone in coda non può scendere di più di una unità per ogni intervallo di tempo (proprietà *skip-free*). In forma matriciale si ha[2]:

$$P = \begin{bmatrix} p_0 + p_1 & p_2 & p_3 & \dots \\ p_0 & p_1 & p_2 & \ddots \\ & p_0 & p_1 & \ddots \\ & & p_0 & \ddots \\ & & & \ddots \end{bmatrix}.$$

La dipendenza solo da $j - i$ ci dà (salvo che nella prima riga) una struttura di *matrice di Toeplitz*: si definisce matrice di Toeplitz una matrice che ha gli elementi uguali lungo le diagonali parallele a quella principale. Inoltre la proprietà *skip-free* ci dà una struttura di *matrice di Hessenberg*: vengono dette matrici di Hessenberg le matrici che hanno tutti gli elementi al di sotto della sottodiagonale nulli.

Un modello più interessante emerge se consideriamo una diversa distribuzione degli arrivi. Supponiamo che gli arrivi, invece di essere indipendenti,

[1] con la notazione $\mathbb{P}[X]$ intendiamo la probabilità che si verifichi l'evento X.

[2] Qui e nel seguito della trattazione, quando in un elemento di una matrice è omesso si intenderà che si tratta di uno zero.

varino in funzione di un'altra catena di Markov che modellizza la situazione dell'"ambiente" (quale può essere il traffico di un'autostrada, o la congestione di una rete di computer). Supponiamo che questa catena φ_k abbia un insieme di stati E (finito) con m elementi. Come stati della catena di Markov scegliamo stavolta le coppie *(numero di persone in coda, stato dell'ambiente)*, cioè $S = \mathbb{N} \times E$: ordinandoli in ordine lessicografico (con la seconda componente che varia più in fretta) abbiamo la matrice a blocchi

$$P = \begin{bmatrix} A_0 + A_1 & A_2 & A_3 & \dots \\ A_0 & A_1 & A_2 & \ddots \\ & A_0 & A_1 & \ddots \\ & & A_0 & \ddots \\ & & & \ddots \end{bmatrix}, \tag{1.3}$$

dove le $A_h \in \mathbb{R}^{m \times m}$ [3] sono date da

$$(A_h)_{ij} = \mathbb{P}\left[\alpha_{k+1} = h, \varphi_{k+1} = j \mid \varphi_k = i\right] \quad \forall k \in \mathbb{N}.$$

Questa matrice ha struttura *Toeplitz a blocchi* e *Hessenberg a blocchi*.

1.4 Il modello M/G/1 e M/G/n

In generale, in letteratura si definisce *coda M/G/1* una coda che ha come matrice di transizione una matrice del tipo

$$\left[\begin{array}{c|ccc} B_0 & B_1 & B_2 & \dots \\ \hline B_{-1} & A_0 & A_1 & \ddots \\ & A_{-1} & A_0 & \ddots \\ & & A_{-1} & \ddots \\ & & & \ddots \end{array}\right] \tag{1.4}$$

(qui, per comodità, abbiamo modificato gli indici delle A_i rispetto alla (1.3) in modo da avere il blocco di indice 0 sulla diagonale). Rispetto all'esempio precedente, abbiamo ammesso che il comportamento della coda nei casi limite in cui essa è vuota possa variare rispetto al caso generale.

Se, più in generale, supponiamo che possano essere servite N persone alla

[3] Indichiamo con $\mathbb{R}^{m \times m}$ l'algebra delle matrici $m \times m$ a coefficienti reali

volta invece che una (*modello M/G/n*), abbiamo una matrice nella forma

$$\left[\begin{array}{c|ccc} B_0 & B_1 & B_2 & \dots \\ \hline B_{-1} & A_0 & A_1 & \ddots \\ \vdots & A_{-1} & A_0 & \ddots \\ B_{-N} & \vdots & A_{-1} & \ddots \\ & A_{-N} & \ddots & \ddots \\ & & A_{-N} & \ddots \\ & & & \ddots \end{array}\right] \tag{1.5}$$

in cui abbiamo N diagonali non nulle anziché una sola al di sotto di quella principale.

1.5 Vettore stazionario di una coda

Segue dall'equazione (1.2) che possiamo modellizzare l'evoluzione di una coda come

$$\begin{cases} x^{(0)} \text{ fissato} \\ x^{(k+1)} = x^{(k)} P \end{cases} \tag{1.6}$$

dove $x_i^{(k)}$ è la probabilità di trovare i persone al tempo k. Si possono verificare sostanzialmente due tipi di comportamento all'infinito: o la coda tende ad aumentare di lunghezza a meno di un insieme con probabilità trascurabile (coda *transiente*), o la probabilità di trovare sempre più persone in coda tende rapidamente a zero (coda *positiva ricorrente*). Più formalmente, una coda è detta *transiente* se da qualunque stato la probabilità di tornare allo stato 0 è strettamente minore di 1 e *ricorrente* se questa è uguale a 1. Inoltre, è detta *positiva ricorrente* se il tempo medio di ritorno allo stato 0 è finito.

Vale la seguente caratterizzazione [2, 6]:

Fatto 1.1. *In una coda irriducibile del tipo dell'equazione* (1.5)*, siano*[4]

$$a = \sum_{i=-N}^{\infty} iA_i\mathbf{1} \tag{1.7}$$

$$b = \sum_{i=-N}^{\infty} iB_i\mathbf{1} \tag{1.8}$$

e α l'unico vettore stocastico tale che $\alpha \sum_{i=-N}^{\infty} A_n = \alpha$ (che esiste in virtù del teorema di Perron–Frobenius [8]). Supponiamo che $a < \infty$ e $b < \infty$.

Allora, la coda è

[4] Con $\mathbf{1}$ si indica comunemente il vettore colonna, di dimensione opportuna, che ha tutti gli elementi uguali a 1.

- *transiente, se* $\alpha a > 0$*;*
- *ricorrente, se* $\alpha a \leq 0$*;*
- *positiva ricorrente, se* $\alpha a \leq 0$ *e* $b < \infty$*.*

Possiamo cercare di visualizzare questo teorema interpretando αa (che è chiamato in letteratura *drift*) come la "crescita media per passo" della coda.

Nel caso ricorrente, si può dimostrare con alcune deboli ipotesi aggiuntive questo risultato cruciale nel contesto del problema che stiamo trattando:

Teorema 1.2. *In una coda positiva ricorrente, irriducibile e non periodica descritta dalle equazioni* (1.6)*, esiste* $\pi \in \ell^1(\mathbb{N})$ *tale che*

$$\lim_{k\to\infty} x_i^{(k)} = \pi_i$$

per ogni $i \in \mathbb{N}$ *e per qualunque scelta del valore iniziale* x_0*. Il vettore* $\pi = (\pi_i)$ *è detto* vettore stazionario *della coda.*

Qui *irriducibile* significa che per ogni $i, j \in \mathbb{N}$ esiste almeno un percorso con probabilità non nulla che porti dallo stato i allo stato j, e *non periodica* significa che le lunghezze di tutti i percorsi siffatti hanno massimo comun divisore uguale a uno: queste ipotesi implicano sostanzialmente che non possiamo "spezzare" la coda in due code più piccole (in due sensi diversi) che possiamo trattare separatamente.

Questa relazione ci dice che, asintoticamente, la probabilità di trovare i persone in coda è uguale a π_i. Calcolando il vettore stazionario quindi possiamo disporre di tutte le informazioni possibili relative al carico della coda, il risultato ci eravamo prefissi di calcolare. Il nostro problema è quindi quello di trovare un algoritmo per il calcolo di π.

Passando al limite la relazione di ricorrenza, otteniamo

$$\pi = \pi P, \tag{1.9}$$

ossia π è una soluzione del sistema lineare omogeneo (infinito) $z(I - P) = 0$. È possibile dimostrare anche che nel caso positivo ricorrente si tratta dell'unica soluzione stocastica di questo sistema (chiaramente, π è stocastico in virtù dell'interpretazione di π_i come probabilità di trovare i persone in coda).

In linea di principio, potremmo calcolare π troncando il vettore alle sue componenti iniziali e risolvendo con metodi standard (ad esempio l'eliminazione di Gauss) il sistema finito (1.9): vedremo infatti nella sezione 3.2 che i termini π_i decadono rapidamente per $i \to \infty$. Tuttavia, il costo computazionale di un algoritmo simile sarebbe eccessivamente elevato, in quanto l'eliminazione di Gauss richiede $O(n^3)$ operazioni aritmetiche elementari (*ops* in breve) su una matrice di dimensione n. In realtà, possiamo utilizzare la struttura Toeplitz e Hessenberg a blocchi delle matrici coinvolte per

giungere a un algoritmo molto più veloce, che impiega soltanto $O(n \log n)$ ops. Il vantaggio pratico di questo algoritmo è evidente, in quanto i valori tipici di n che si incontrano nelle applicazioni possono arrivare fino a 10^5. Descriveremo questo algoritmo nei capitoli successivi, dopo aver introdotto gli strumenti necessari.

Capitolo 2

Algoritmi per matrici di Toeplitz

2.1 Trasformata discreta di Fourier

Sia n un intero positivo fissato. Prendiamo un polinomio di grado minore di n, $a(z) = \sum_{i=0}^{n-1} a_i z^i \in \mathbb{C}[z]$ e ω una radice n-esima primitiva dell'unità e Consideriamo l'applicazione lineare

$$\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{n-1} \end{bmatrix} \mapsto \begin{bmatrix} a(1) \\ a(\omega) \\ \vdots \\ a(\omega^{n-1}) \end{bmatrix},$$

cioè

$$\begin{bmatrix} \textit{coefficienti} \\ \textit{del polinomio} \end{bmatrix} \mapsto \begin{bmatrix} \textit{Valori assunti dal polinomio} \\ \textit{nelle radici } n\textit{-esime di } 1 \end{bmatrix}.$$

Si può verificare che essa è invertibile; la sua inversa, cioè l'operazione di *interpolazione* di un polinomio di grado minore di n conoscendo i valori che assume nelle radici n-esime dell'unità, è detta *trasformata discreta di Fourier*, o DFT. L'operazione di valutazione che abbiamo considerato inizialmente è invece detta DFT *inversa* o IDFT.

La trasformata discreta di Fourier ha proprietà estremamente utili dal punto di vista computazionale, che la rendono adatta a svolgere in modo veloce ed efficiente calcoli che coinvolgono i polinomi. In particolare, ricordiamo questi fatti:

Teorema 2.1 (Proprietà della DFT **[1]).**

1. DFT *e* IDFT *si calcolano con* $O(n \log n)$ *operazioni aritmetiche, invece che in* $O(n^2)$ *come una generica applicazione lineare, con un algorit-*

mo noto come Fast Fourier Transform, e la costante implicita nella notazione $O(\cdot)$ è piccola.

2. DFT *e* IDFT *sono (a meno di un fattore di scala $\frac{1}{n}$) isometrie per la norma-2. In quanto tali, non amplificano un eventuale errore numerico (in norma-2) sui dati di ingresso.*

Un'applicazione che illustra l'uso della DFT è questo algoritmo per il prodotto di due polinomi:

1. dati a_i, b_i coefficienti di $a(z) = \sum_{i=0}^{r} a_i z^i$ e $b(z) = \sum_{i=0}^{s} b_i z^i$:
2. scegliamo $n > \deg a + \deg b$ per essere sicuri di ricostruire correttamente il polinomio prodotto;
3. calcoliamo IDFT(a) e IDFT(b), cioè i valori assunti da $a(z)$ e $b(z)$ nelle radici n-esime dell'unità;
4. $c(z) = a(z)b(z)$ assume i valori $a(\omega^i)b(\omega^i)$ nei punti ω^i: calcoliamo i prodotti termine a termine $c(\omega^i) = a(\omega^i)b(\omega^i)$;
5. ricostruiamo i coefficienti di $c(z)$ con una DFT: $c = \mathrm{DFT}(c(\omega^i))$.

Il costo computazionale di questo algoritmo è pari a quello delle trasformate di Fourier, vale a dire $O(n \log n)$ ops; se invece avessimo calcolato algebricamente il prodotto utilizzando la proprietà distributiva, il costo sarebbe stato un più elevato $O(n^2)$.

Osserviamo anche che moltiplicando per una potenza di z opportuna è semplice adattare questo algoritmo ai *polinomi di Laurent*, cioè espressioni del tipo $\sum_{i=-n}^{m} a_i z^i$, con $n, m \in \mathbb{N}$, in cui ammettiamo anche potenze negative dell'indeterminata.

2.2 Prodotto Toeplitz–vettore

Se indichiamo con questa notazione un prodotto matrice di Toeplitz–vettore:

$$\begin{bmatrix} a_0 & a_{-1} & \dots & a_{-n} \\ a_1 & a_0 & \ddots & a_{-n+1} \\ \vdots & \ddots & \ddots & \vdots \\ a_n & a_{n-1} & \dots & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} a_0 b_0 + a_{-1} b_1 + \cdots + a_{-n} b_n \\ a_1 b_0 + a_0 b_1 + \cdots + a_{-n+1} b_n \\ \vdots \\ a_n b_0 + a_{n-1} b_1 + \cdots + a_0 b_n \end{bmatrix},$$

possiamo notare che i coefficienti del vettore prodotto altro non sono che i coefficienti $c_0, c_1, \dots c_n$ del prodotto tra polinomi (di Laurent)

$$\sum_{i=-2n}^{2n} c_i z^i = \left(\sum_{i=-n}^{n} a_i z^i \right) \left(\sum_{i=0}^{n} b_i z^i \right).$$

Quindi, per ottenere il prodotto Toeplitz–vettore tutto quello che dobbiamo fare è calcolare il prodotto dei due polinomi di Laurent ed estrarre i coefficienti di cui abbiamo bisogno. Con l'ausilio della DFT, quest'operazione richiede solamente $O(n \log n)$ operazioni.

Notiamo infine che l'algoritmo si può adattare facilmente (trasponendo tutte le matrici e i vettori coinvolti) anche al prodotto vT tra un vettore riga e una matrice di Toeplitz. Questa connessione tra matrici di Toeplitz e polinomi non è casuale ma ha radici più profonde, che esamineremo in parte nel capitolo successivo.

2.3 Inversione di matrici di Toeplitz triangolari

Descriviamo qui un algoritmo per calcolare velocemente l'inversa di una matrice di Toeplitz triangolare di dimensione potenza di due. Considereremo solo il caso delle triangolari inferiori, in quanto l'algoritmo si adatta banalmente alle triangolari superiori: poiché inversione e trasposizione commutano, ci basta invertire la trasposta di una triangolare superiore e poi trasporre la matrice ottenuta.

Notiamo innanzitutto che l'inversa resta ancora una matrice di Toeplitz:

Teorema 2.2. *L'inversa di una matrice di Toeplitz triangolare inferiore (risp. superiore), quando esiste, è un'altra matrice di Toeplitz triangolare inferiore (risp. superiore).*

Dimostrazione. È semplice verificare che la classe delle matrici di Toeplitz triangolari inferiori, che chiameremo $\mathcal{T}$, è chiusa per somma e prodotto (i.e. è una sotto-algebra di $\mathbb{R}^{m \times m}$). Ora, sia $T \in \mathcal{T}$ non singolare. Per il teorema di Cayley–Hamilton, esiste un polinomio a coefficienti reali a_i che si annulla in T: $\sum_{i=0}^{n} a_i T^i = 0$, e possiamo supporre $a_0 \neq 0$ (altrimenti possiamo raccogliere una potenza di T e ridurci a un polinomio di grado inferiore). Allora, moltiplicando la relazione per T^{-1} e isolando il termine in T^{-1} otteniamo

$$T^{-1} = -\frac{1}{a_0} \sum_{i=1}^{n} a_i T^{i-1}.$$

Ma il termine a destra dell'uguale appartiene evidentemente all'algebra $\mathcal{T}$: quindi anche $T^{-1} \in \mathcal{T}$.

La dimostrazione per le triangolari superiori procede in modo completamente analogo. □

L'idea di questo algoritmo – che non è dissimile da quella dell'algoritmo per la Fast Fourier Transform – è quella di risolvere il problema in dimensione 2^{k-1} e poi "raddoppiare" la dimensione utilizzando il risultato già calcolato.

Per invertire una matrice di Toeplitz triangolare inferiore A di dimensione 2^k (che supponiamo non singolare), la dividiamo in blocchi di dimensione

2^{k-1}:

$$A = \left[\begin{array}{c|c} T & 0 \\ \hline S & T \end{array}\right]$$

(i due blocchi denominati T sono uguali perché la matrice è di Toeplitz). L'inversa di questa matrice si può calcolare utilizzando a blocchi la consueta formula per l'inversione di matrici 2×2:

$$A^{-1} = \left[\begin{array}{c|c} T^{-1} & 0 \\ \hline -T^{-1}ST^{-1} & T^{-1} \end{array}\right].$$

Ora, T è una matrice di Toeplitz di dimensione 2^{k-1}, quindi possiamo invertirla applicando ricorsivamente l'algoritmo (una volta arrivati a dimensione 1, l'inversione è banale). Per quanto riguarda il prodotto $-T^{-1}ST^{-1}$, per conoscere A^{-1} ci basta calcolarne la prima colonna (in quanto anche A^{-1} è di Toeplitz), per cui possiamo calcolarlo con due prodotti Toeplitz–vettore con l'algoritmo illustrato qui sopra.

Il calcolo della complessità computazionale in questo caso è leggermente complicato dalla ricorsione, ma si può dimostrare che anche questo algoritmo richiede $O(n \log n)$ ops.

2.4 Estensione alle matrici a blocchi

Gli algoritmi trattati in questo capitolo si estendono facilmente alle matrici a blocchi: poiché in essi si richiedono solamente somme e prodotti, possiamo eseguire formalmente le stesse operazioni anche nel caso in cui gli "elementi" siano blocchi $m \times m$ (utilizzando la struttura di anello delle matrici e l'equivalenza tra operazioni *element-wise* e *block-wise*). L'unico punto che richiede un po' di attenzione è il passo $k = 0$ dell'algoritmo per l'inversione: dobbiamo calcolare l'inversa di una matrice di dimensione un blocco $m \times m$. Però, in questo caso, se la matrice triangolare con cui stiamo lavorando è invertibile lo è anche la sua sottomatrice principale di testa $m \times m$, quindi non abbiamo problemi di singolarità.

Il costo computazionale di questi algoritmi per matrici a blocchi assume allora un aspetto leggermente più complicato: se abbiamo n blocchi $m \times m$, dobbiamo:

- calcolare la DFT di lunghezza n di matrici $m \times m$: questo si fa con $O(n \log n)$ operazioni di somma di matrici e prodotto per scalare, ognuna delle quali richiede m^2 operazioni aritmetiche di base;
- calcolare i prodotti tra matrici $A(\omega^i)B(\omega^i)$ dell'algoritmo per il prodotto di polinomi: si tratta di n prodotti di matrici $m \times m$, ognuno dei quali si fa in $O(m^3)$ ops.

Quindi il costo complessivo di ognuno di questi algoritmi nel caso a blocchi è $O(m^2 n \log n + m^3 n)$ e a seconda dei valori assunti da m e n il termine dominante può essere il primo o il secondo.

Capitolo 3

Algebra di Wiener e fattorizzazione canonica

3.1 Algebra di Wiener

Chiamiamo *classe di Wiener* l'insieme $\mathcal{W}$ delle serie di Laurent $\sum_{i=-\infty}^{+\infty} A_i z^i$ a valori in $\mathbb{C}^{m\times m}$ tali che

$$\sum_{i=-\infty}^{+\infty} |A_i| < \infty.$$

L'espressione $|A|$, con A matrice $m \times m$, indica qui la matrice che ha per elementi i valori assoluti degli elementi di A; pertanto la somma è a valori matriciali e intendiamo che essa deve essere finita elemento per elemento. Tuttavia questa definizione classica è equivalente a chiedere che $\sum_{i=-\infty}^{+\infty} \|A_i\| < \infty$ per una qualunque delle usuali norme matriciali.

È semplice verificare che la classe di Wiener è un'algebra; al suo interno abbiamo la seguente caratterizzazione degli elementi invertibili:

Fatto 3.1. *$A(z) \in \mathcal{W}$ è invertibile se e solo se è non singolare per tutti gli z con $|z| = 1$.*

Tale caratterizzazione non deve sorprendere: se pensiamo alla classe di Wiener come all'insieme delle funzioni $\mathbb{C} \to \mathbb{C}^{m\times m}$ analitiche in un intorno della circonferenza unitaria, questa caratterizzazione è l'estensione alle matrici dell'usuale enunciato di analisi che afferma che se $f(z) \neq 0$ per una funzione analitica allora $\frac{1}{f(z)}$ esiste ed è analitica sullo stesso dominio.

Richiamiamo l'attenzione anche su due sottoclassi particolari della classe di Wiener, che utilizzeremo in futuro:

- $\mathcal{W}^+$ è la classe delle serie di potenze usuali, cioè $\mathcal{W}^+ = \{A(z) \in \mathcal{W} \text{ t.c. } A(z) = \sum_{i=0}^{+\infty} A_i z^i\}$;
- $\mathcal{W}^-$ è la classe delle serie di potenze in $\frac{1}{z}$, cioè $\mathcal{W}^- = \{A(z) \in \mathcal{W} \text{ t.c. } A(z) = \sum_{i=0}^{+\infty} A_{-i} z^{-i}\}$.

3.2 ε-grado e grado numerico di serie in $\mathcal{W}$

Sia $A(z)$ un elemento di $\mathcal{W}$. Poiché la serie $\sum_{i=-\infty}^{+\infty} |A_i|$ è convergente, per ogni $\varepsilon > 0$ esistono due interi positivi r, s tali che

$$\sum_{i<-r} |A_i| + \sum_{i>s} |A_i| < \varepsilon.$$

Quando ciò si verifica, diremo che $A(z)$ ha *ε-grado* minore o uguale a (r, s). Se ε è uguale alla precisione di macchina del computer su cui dovremo eseguire i calcoli, diremo che $A(z)$ ha *grado numerico* minore o uguale a (r, s). La motivazione di questa definizione è evidente: nell'aritmetica *floating point* del computer i termini delle due "code" della serie sono trascurabili in quanto minori della precisione di macchina, quindi nei nostri calcoli possiamo considerare a tutti gli effetti $A(z)$ come un polinomio di Laurent di grado (r, s).

3.3 Matrici associate all'algebra di Wiener

Associamo a una serie dell'algebra di Wiener $A(z) = \sum_{i=-\infty}^{+\infty} A_i z^i$ due matrici infinite, a indici rispettivamente in $\mathbb{N}$ (matrici *semi-infinite*) e $\mathbb{Z}$ (matrici *bi-infinite*):

$$T_\infty[A] := \begin{bmatrix} A_0 & A_1 & A_2 & \dots \\ A_{-1} & A_0 & A_1 & \ddots \\ A_{-2} & A_{-1} & A_0 & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix},$$

$$T_{\pm\infty}[A] := \begin{bmatrix} \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & A_0 & A_1 & A_2 & \ddots \\ \ddots & A_{-1} & A_0 & A_1 & \ddots \\ \ddots & A_{-2} & A_{-1} & A_0 & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$

cioè, in entrambe le matrici, l'elemento di posto (i, j) è A_{j-i}. Si noti che tali matrici rappresentano effettivamente operatori continui in ℓ^1, in quanto la condizione di Wiener ne garantisce la limitatezza.

La seconda di queste mappe, $A(z) \mapsto T_{\pm\infty}[A]$, fornisce un isomorfismo di algebre con l'algebra di Wiener: possiamo verificare infatti che $A(z)B(z) = C(z) \iff T_{\pm\infty}[A]T_{\pm\infty}[B] = T_{\pm\infty}[C]$ notando che nel prodotto di serie e nel prodotto di matrici le operazioni che dobbiamo svolgere sono formalmente le stesse. Alternativamente, possiamo arrivare allo stesso risultato notando che la $T_{\pm\infty}[A]$ è la matrice associata all'applicazione lineare $X \mapsto AX$ nell'algebra di Wiener con la base dei monomi.

3.4 Fattorizzazione canonica e di Wiener–Hopf

Saremo interessati a studiare all'interno dell'algebra di Wiener alcuni modi particolari di fattorizzare gli elementi. Una *fattorizzazione di Wiener–Hopf* è una scomposizione del tipo

$$A(z) = U(z)D(z)L(z)$$

dove:

- $U(z)$ è un elemento di $\mathcal{W}^+$ tale che $\det U(z) \neq 0$ per $|z| \leq 1$;
- $L(z)$ è un elemento di $\mathcal{W}^-$ tale che $\det L(1/z) \neq 0$ per $|z| \leq 1$ (queste condizioni in realtà equivalgono a supporre che $U(z)$ e $L(z)$ siano invertibili rispettivamente in $\mathcal{W}^+$ e $\mathcal{W}^-$);
- $D(z)$ è una matrice diagonale della forma

$$\begin{bmatrix} z^{\kappa_1} & & & \\ & z^{\kappa_2} & & \\ & & \ddots & \\ & & & z^{\kappa_m} \end{bmatrix}$$

  con i κ_i interi.

I κ_i sono detti *indici parziali*. Nel caso particolare in cui le matrici trattate sono di dimensione 1 (e quindi l'algebra di Wiener è a valori in $\mathbb{C}$), si ha che l'indice parziale coincide con l'usuale nozione di indice di avvolgimento (o *winding number*) intorno all'origine.

Si può dimostrare [4] che tutte gli elementi invertibili all'interno dell'algebra di Wiener ammettono una fattorizzazione di Wiener–Hopf.

Chiameremo *fattorizzazione canonica* una fattorizzazione di Wiener–Hopf in cui $D(z)$ sia uguale all'identità. Se la leggiamo attraverso l'isomorfismo della sezione 3.3, una fattorizzazione canonica assume l'aspetto di una fattorizzazione UL di matrici bi-infinite:

$$\begin{bmatrix} \ddots & \ddots & \ddots & \ddots & \ddots \\ \ddots & A_0 & A_1 & A_2 & \ddots \\ \ddots & A_{-1} & A_0 & A_1 & \ddots \\ \ddots & A_{-2} & A_{-1} & A_0 & \ddots \\ \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix} = \begin{bmatrix} \ddots & \ddots & \ddots & \ddots & \ddots \\ & U_0 & U_1 & U_2 & \ddots \\ & & U_0 & U_1 & \ddots \\ & & & U_0 & \ddots \\ & & & & \ddots \end{bmatrix} \begin{bmatrix} \ddots & & & & \\ \ddots & L_0 & & & \\ \ddots & L_{-1} & L_0 & & \\ \ddots & L_{-2} & L_{-1} & L_0 & \\ \ddots & \ddots & \ddots & \ddots & \ddots \end{bmatrix}$$

cioè, la scomposizione di una matrice bi-infinita nel prodotto di una triangolare superiore per una triangolare inferiore. Per come sono disposti gli

zeri nelle due matrici, questa fattorizzazione vale anche per le matrici semi-infinite, per cui a priori non avremmo un isomorfismo:

$$\begin{bmatrix} A_0 & A_1 & A_2 & \dots \\ A_{-1} & A_0 & A_1 & \ddots \\ A_{-2} & A_{-1} & A_0 & \ddots \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix} = \begin{bmatrix} U_0 & U_1 & U_2 & \dots \\ & U_0 & U_1 & \ddots \\ & & U_0 & \ddots \\ & & & \ddots \end{bmatrix} \begin{bmatrix} L_0 & & & \\ L_{-1} & L_0 & & \\ L_{-2} & L_{-1} & L_0 & \\ \vdots & \ddots & \ddots & \ddots \end{bmatrix}.$$

Chiameremo inoltre *fattorizzazione canonica debole* una fattorizzazione $A = UL$ tale che $U(z) \in \mathcal{W}^+$ sia non singolare per $|z| < 1$ e $L(z) \in W^-$ sia non singolare per $|z| > 1$: cioè, indeboliamo leggermente le ipotesi della fattorizzazione canonica in modo da ammettere che U e L (e quindi anche A) possano essere singolari sulla circonferenza unitaria. Quest'ultimo caso ricopre nella nostra trattazione un ruolo importante in quanto dimostreremo che la matrice di Toeplitz associata a una coda M/G/1 ammette una fattorizzazione canonica debole.

3.5 Esistenza della fattorizzazione canonica debole per una coda M/G/1

In questa sezione dimostreremo che la matrice $I - A(z)$ (con $A(z) = \sum_{i=-1}^{\infty} A_i z^i$, gli A_i come nella (1.4)) associata a una coda M/G/1 ammette una fattorizzazione canonica debole. La dimostrazione si può estendere con idee simili ma qualche complicazione in più anche nel caso delle code M/G/n.

Ricordiamo che in una catena M/G/1 gli stati in generale sono della forma (k, i), dove $k \in \mathbb{N}$ è il numero di persone in coda ("livello"), e $i \in E$ è uno stato della catena ambiente. Per brevità di notazione, scriveremo $(k, i) \rightsquigarrow (h, j)$ per indicare che la coda passa dallo stato (k, i) allo stato (h, j) in tempo finito, e durante questo percorso (h, j) è il primo stato che viene raggiunto al livello h.

Definiamo la matrice $G \in \mathbb{R}^{m \times m}$ come $G_{ij} = \mathbb{P}\left[(h+1, i) \rightsquigarrow (h, j)\right]$ per un qualunque $h > 0$ (per la proprietà di omogeneità della coda, la probabilità non dipende dalla scelta di h). Per la proprietà di positiva ricorrenza, la probabilità di tornare al livello h in tempo finito partendo da qualunque stato è pari a 1, quindi $\sum_{j=1}^{m} G_{ij} = 1$, ossia G è stocastica.

Definiamo analogamente anche $G^{(k)}$ come $G^{(k)}_{ij} = \mathbb{P}\left[(h+k, i) \rightsquigarrow (h, j)\right]$, cioè la probabilità di passare da $h + k$ a h persone in coda muovendosi contemporaneamente dallo stato i allo stato j dell'ambiente. Di nuovo, la probabilità non dipende dalla scelta di $h > 0$.

Dimostriamo preliminarmente alcuni risultati:

Lemma 3.2. *Per ogni $n \geq 1$ si ha*

$$G^{(n)} = G^n.$$

Dimostrazione. Per la proprietà *skip-free*, per passare dal livello $h+2$ al livello h dobbiamo passare necessariamente per il livello $h+1$: se decomponiamo la probabilità a seconda di quale sia il primo stato $(h+1,r)$ visitato al livello $h+1$ abbiamo

$$G_{ij}^{(2)} = \sum_{r\in E} \mathbb{P}\left[(h+2,i) \rightsquigarrow (h+1,r) \text{ seguita da } (h+1,r) \rightsquigarrow (h,j)\right] = \sum_{r\in E} G_{ir}G_{rj}$$

ossia, in forma matriciale, $G^{(2)} = G^2$. In modo analogo possiamo dimostrare per induzione che $G^{(n)} = G^n$, in quanto decomponendo sul primo stato r visitato al livello $h+n-1$ abbiamo

$$G_{ij}^{(n)} = \sum_{r\in E} G_{ir}G_{rj}^{(n-1)}$$

e quindi $G^{(n)} = GG^{(n-1)}$. □

Lemma 3.3. *La matrice G definita in precedenza soddisfa*

$$G = \sum_{i=-1}^{\infty} A_i G^{i+1} = A(G)G. \tag{3.1}$$

Dimostrazione. Decomponiamo le probabilità di transizione dal livello h al livello $h-1$ a seconda di quale sia lo stato visitato all'istante successivo dalla catena di Markov. Per compiere una transizione dal livello h al livello $h-1$, la coda può passare dal livello h al livello $h-1$ direttamente al primo passaggio (con probabilità $(A_{-1})_{ij}$, a seconda degli stati dell'ambiente coinvolti), oppure salire di k livelli (per un qualunque $k \in \mathbb{N}$, con probabilità $(A_k)_{ij}$) e poi scendere di nuovo in tempo finito al livello $h-1$ (con probabilità $G^{(k+1)}$). Quindi, in formule,

$$G = A_{-1} + A_0G + A_1G^2 + A_2G^3 + \ldots = A(G)G.$$ □

Con questi risultati possiamo ora dimostrare l'esistenza della fattorizzazione canonica debole:

Teorema 3.4. *La serie $I - A(z)$ ammette una fattorizzazione canonica debole del tipo*

$$I - A(z) = U(z)L(z)$$

con

$$L(z) = I - Gz^{-1},$$
$$U(z) = I - \sum_{i=0}^{n} A_i^* z^i,$$

dove $A_n^* = \sum_{i=n}^{\infty} A_i G^{i-n}$.

Dimostrazione. Innanzitutto notiamo che i termini A_i^* sono finiti perché le potenze di G sono stocastiche e quindi tutte limitate uniformemente (G stocastica implica $\rho(G) = \|G\|_1 = 1$).

Utilizzando la (3.1) e il fatto che $A_i = A_i^* - A_{i+1}^* G$ per tutti gli $i \geq 0$, la fattorizzazione si verifica esplicitamente:

$$\begin{aligned}(UL)_{-1} &= A_0^* G - G = \sum_{i=0}^{\infty} A_i G^{i+1} - G = -A_{-1}, \\ (UL)_0 &= I - A_0^* + A_1^* G = I - A_0, \\ (UL)_k &= -A_k^* + A_{k+1}^* G = -A_k \qquad \forall k > 0.\end{aligned}$$

Ora ci restano da verificare le proprietà di non singolarità della U e della L. La L è evidentemente non singolare per $|z| > 1$ perché vale la formula di inversione

$$(I - Gz^{-1})^{-1} = \sum_{i=0}^{\infty} z^{-i} G^i$$

che converge in quanto $\|G\|_1 = 1$. La non singolarità della U per $|z| < 1$ è più complessa da dimostrare, ma può essere provata facendo uso del seguente risultato [6]:

Teorema 3.5. *In una coda M/G/1, se il drift è negativo allora* $a(z) = \det(z(I - A(z)))$ *ha esattamente* m *zeri (contati con molteplicità) all'interno del disco unitario chiuso.*

Infatti, a ogni autovettore v di G corrisponde necessariamente uno zero del determinante di $z(I - A(z))$, in quanto per la (3.1) si ha

$$\lambda v = Gv = \sum_{i=-1}^{\infty} A_i G^{i+1} v = \sum_{i=-1}^{\infty} A_i \lambda^{i+1} v = \lambda A(\lambda) v.$$

Se G ha m autovettori distinti, gli autovalori corrispondenti devono stare all'interno del disco unitario perché $\rho(G) = 1$: per il teorema appena enunciato si tratta di tutti i punti in cui $z(I - A(z))$ è singolare all'interno del disco unitario, quindi tutti i punti di singolarità di $U(z)$ (che sono anche punti di singolarità per $I - A(z) = U(z)L(z)$) devono stare all'*esterno* del disco unitario.

I casi in cui $\det(zI - G)$ ha zeri con molteplicità $\kappa > 1$ si possono ricondurre a quello precedente con argomenti di continuità oppure si possono trattare direttamente notando che a zeri di molteplicità κ corrispondono blocchi di Jordan di G di dimensione κ, e la somma delle dimensioni dei blocchi di Jordan per G fa esattamente m. □

Capitolo 4

Algoritmo per il calcolo del vettore stazionario

4.1 Tecnica di shift

L'idea che cercheremo di applicare per il calcolo del vettore stazionario passa attraverso un algoritmo per calcolare la fattorizzazione canonica di un elemento dell'algebra di Wiener. Tuttavia, la serie $\varphi(z) = I - A(z)$ a cui siamo interessati non ammette una fattorizzazione canonica: infatti, l'esistenza di una fattorizzazione di Wiener–Hopf implicherebbe la non singolarità della serie per $|z| = 1$, ma per la stocasticità della matrice A si ha

$$\sum_{i=-N}^{\infty} A_i \mathbf{1} = \mathbf{1}$$

e quindi $(I - A(1))\mathbf{1} = 0$. Possiamo però modificare opportunamente la serie in modo da eliminare la singolarità, e ricavare dalla fattorizzazione canonica di questa serie modificata una fattorizzazione canonica debole per la serie di partenza. Per farlo utilizzeremo le ipotesi aggiuntive che $\sum_{i=0}^{\infty} iA_i < \infty$ (già fatta nel teorema 1.1) e che $z = 1$ sia l'unica singolarità di $\varphi(z) = I - A(z)$ sulla circonferenza unitaria (ipotesi che può essere derivata con la teoria di Perron–Frobenius [8] dalle ipotesi del teorema 1.2).

Scegliamo $u \in \mathbb{R}^m$ un vettore tale che $u^T\mathbf{1} = 1$, e sia $Q = \mathbf{1}u^T$. Definiamo la serie

$$\tilde{\varphi}(z) = \varphi(z)(I - z^{-1}Q)^{-1}.$$

Poiché $Q^i = Q \ \forall i \geq 1$ per come è stato scelto u, abbiamo $(I - z^{-1}Q)^{-1} = I + Q\sum_{i=1}^{\infty} z^{-i}$: quindi la serie $\tilde{\varphi}(z)$ ha come coefficienti

$$\tilde{\varphi}_j = \varphi_j + \left(\sum_{i=j+1}^{\infty} \varphi_i\right) Q. \tag{4.1}$$

Poiché $\left(\sum_{i=-N}^{\infty} \varphi_i\right) Q = (I - A(1))\,\mathbf{1}u^T = 0$, $\tilde{\varphi}_{-N}$ è il primo termine non nullo di $\tilde{\varphi}(z)$. Inoltre si ha

$$\begin{aligned}\sum_{j=-N}^{\infty} |\tilde{\varphi}_j| &\leq \sum_{j=-N}^{\infty} |\tilde{\varphi}_j| + \sum_{j=-N}^{\infty} \left(\sum_{i=j+1}^{\infty} |\varphi_i| \right) |Q| \\ &\leq \sum_{j=-N}^{\infty} |\tilde{\varphi}_j| + \left(\sum_{j=-N}^{\infty} |\tilde{\varphi}_j| \right)^2 |Q|\end{aligned}$$

così $\tilde{\varphi}$ sta nell'algebra di Wiener, e

$$\det \tilde{\varphi}(z) = \frac{z}{z-1} \det \varphi(z),$$

quindi la singolarità in 1 di $(I - z^{-1}Q)^{-1}$ compensa lo zero di $A(z)$ (che avevamo supposto essere semplice): perciò la serie $\tilde{\varphi}(z)$ è non singolare su tutta la circonferenza unitaria.

Ora, se $\tilde{\varphi} = \tilde{U}\tilde{L}$ è una fattorizzazione canonica, allora

$$\varphi(z) = \tilde{U}\left(\tilde{L}(I - z^{-1}Q)\right)$$

è una fattorizzazione canonica debole per φ: quindi i coefficienti delle due fattorizzazioni sono legati da

$$\begin{cases} U_i = \tilde{U}_i. \\ L_i = \tilde{L}_i - L_{i-1}Q. \end{cases} \tag{4.2}$$

Notiamo anche che, in conseguenza del fatto che $\sum_{i=-N}^{\infty} \varphi_i \mathbf{1} = 0$ (e che $Q = \mathbf{1}u^T$) la (4.1) diventa

$$\tilde{\varphi}_j = \varphi_j - \left(\sum_{i=-N}^{j} \varphi_j \right) Q = \varphi_j(I - Q) - \varphi_{j-1} + \tilde{\varphi}_{j-1}$$

da cui possiamo calcolare ricorsivamente i termini di $\tilde{\varphi}(z)$ a partire da $\tilde{\varphi}_{-N}$ con un costo computazionale di $O(m^3)$ per ogni termine.

4.2 Calcolo della fattorizzazione canonica

Ricaveremo in questa sezione un algoritmo per calcolare in modo veloce la fattorizzazione canonica di una serie $\varphi(z) = \sum_{i=-N}^{+\infty} A_i z^i \in \mathcal{W}$, nell'ipotesi in cui essa esista e N sia piccolo rispetto al grado numerico di φ.

Per l'isomorfismo esibito, il problema è equivalente a calcolare una fattorizzazione UL della matrice $T_{\pm\infty}[A]$. Poniamo $q = N + 1$ e modifichiamo la struttura delle matrici bi-infinite coinvolte nella fattorizzazione in modo da

raggruppare i blocchi $m \times m$ in blocchi più grandi di dimensione $mq \times mq$, in questo modo:

$$T_{\pm\infty}[B] = \left[\begin{array}{c|ccc|ccc|c} \ddots & & \ddots & & & \ddots & & \ddots \\ \hline & B_0 & \dots & B_N & B_{N+1} & \dots & B_{2N+2} & \\ \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots \\ & B_{-N} & \dots & B_0 & B_1 & \dots & B_{N+1} & \\ \hline & B_{-N-1} & \dots & B_{-1} & B_0 & \dots & B_N & \\ \ddots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \ddots \\ & B_{-2N-2} & \dots & B_{-N-1} & B_{-N} & \dots & B_0 & \\ \hline \ddots & & \ddots & & & \ddots & & \ddots \end{array}\right]$$

$$= \left[\begin{array}{c|cc|c} \ddots & \ddots & \ddots & \ddots \\ \hline \ddots & \mathcal{B}_0 & \mathcal{B}_1 & \ddots \\ \hline \ddots & \mathcal{B}_{-1} & \mathcal{B}_0 & \ddots \\ \hline \ddots & \ddots & \ddots & \ddots \end{array}\right].$$

Come sopra, indicheremo con lettere corsive i "blocchi composti" ottenuti dopo questa operazione di reblocking. La relazione matriciale $I - A = UL$, letta in termini delle serie associate alle matrici ribloccate, diventa allora

$$I - \sum_{i=-1}^{+\infty} \mathcal{A}_i z^i = \left(\sum_{i=0}^{+\infty} \mathcal{U}_i z^i\right) \left(\mathcal{L}_0 + \mathcal{L}_{-1} z^{-1}\right). \tag{4.3}$$

Vogliamo scrivere l'inversa di questa relazione: se definiamo $\mathcal{H}_i$ e $\mathcal{V}_i$ come

$$I - \mathcal{A}(z) = \sum_{i=-\infty}^{+\infty} \mathcal{H}_i z^i, \quad \mathcal{U}^{-1}(z) = \sum_{i=0}^{+\infty} \mathcal{V}_i z^i,$$

invertendo termine a termine la (4.3) otteniamo la formula

$$\sum_{i=-\infty}^{+\infty} \mathcal{H}_i z^i = \mathcal{L}_0^{-1} \left(\sum_{i=0}^{+\infty} \left(-\mathcal{L}_{-1}\mathcal{L}_0^{-1}\right)^i z^{-i}\right) \sum_{i=0}^{+\infty} \mathcal{V}_i z^i$$

in cui abbiamo usato l'identità algebrica $(1-x)^{-1} = \sum_{i=0}^{+\infty} x^i$ per invertire $\left(\mathcal{L}_0 + \mathcal{L}_{-1} z^{-1}\right)$: la convergenza della serie sulla circonferenza unitaria è assicurata dal fatto che per le ipotesi della fattorizzazione canonica la $\mathcal{L}(z)$ è

non singolare per $|z| \geq 1$. Calcoliamo il termine di grado 0 della relazione precedente:

$$\mathcal{H}_0 = \mathcal{L}_0^{-1} \sum_{i=0}^{\infty} \left(-\mathcal{L}_{-1}\mathcal{L}_0^{-1}\right)^i \mathcal{V}_i.$$

Ritorniamo ora all'interpretazione in blocchi $m \times m$: le matrici qui considerate sono matrici di dimensione $mq \times mq$, che possiamo pensare come composte da q^2 blocchi $m \times m$. Isoliamo i blocchi corrispondenti all'ultima riga moltiplicando a sinsitra per $\begin{pmatrix} 0 & 0 & \cdots & I_m \end{pmatrix}$, cioè un "vettore riga a blocchi" di dimensione $m \times mq$ con i primi $q-1$ blocchi nulli e l'identità nell'ultimo blocco:

$$\begin{bmatrix} 0 & 0 & \cdots & I \end{bmatrix} \mathcal{L}_0\mathcal{H}_0 = \begin{bmatrix} 0 & 0 & \cdots & I \end{bmatrix} \sum_{i=0}^{\infty} \left(-\mathcal{L}_{-1}\mathcal{L}_0^{-1}\right)^i \mathcal{V}_i. \tag{4.4}$$

Ora,

$$\mathcal{L}_{-1} = \begin{bmatrix} 0 & L_N & \dots & L_{-1} \\ & 0 & \ddots & \vdots \\ & & \ddots & L_N \\ & & & 0 \end{bmatrix}$$

ha l'ultima riga di blocchi nulla, quindi nel membro di destra della (4.4) tutti i termini della sommatoria sono nulli tranne quello con $i = 0$. Abbiamo infine

$$\begin{bmatrix} L_N & L_{N-1} & \dots & I \end{bmatrix} \mathcal{H}_0 = \begin{bmatrix} 0 & 0 & \cdots & I \end{bmatrix} \mathcal{V}_0 = \begin{bmatrix} 0 & 0 & \cdots & U_0^{-1} \end{bmatrix}.$$

Quindi abbiamo un sistema $qm \times qm$ da cui possiamo ricavare i coefficienti della L:

$$\begin{bmatrix} U_0L_N & U_0L_{N-1} & \dots & U_0 \end{bmatrix} \mathcal{H}_0 = \begin{bmatrix} 0 & 0 & \cdots & I \end{bmatrix}.$$

Se q è "piccolo" rispetto al grado numerico dei vettori coinvolti siamo in grado di risolvere velocemente (in $O(q^3m^3)$ ops) questo sistema con l'eliminazione gaussiana e ottenere quindi la L della fattorizzazione canonica. Poi la U si calcola facilmente risolvendo il sistema triangolare di Toeplitz

$$\begin{bmatrix} A_0 & A_1 & \dots \end{bmatrix} L = \begin{bmatrix} U_0 & U_1 & \dots \end{bmatrix}$$

con le tecniche della sezione 2.3. Il fatto che i vettori a blocchi (A_i) e (U_i) siano infiniti non costituisce un problema: su un computer, possiamo troncarli senza introdurre ulteriori errori alla lunghezza data dalla più piccola potenza di 2 maggiore del grado numerico di $A(z)$.

4.3 Calcolo dei coefficienti centrali di φ^{-1}

Per poter applicare l'algoritmo descritto nella sezione precedente dobbiamo prima calcolare esplicitamente la matrice

$$\mathcal{H}_0 = \begin{bmatrix} H_0 & H_1 & \dots & H_N \\ H_{-1} & H_0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & H_1 \\ H_{-N} & \dots & H_{-1} & H_0 \end{bmatrix},$$

dove gli H_i sono i coefficienti della serie inversa di $\varphi(z)$.

Per effettuare questo calcolo, ricorriamo nuovamente alla DFT con questo algoritmo [3]:

1. scegliamo n potenza di due "sufficientemente grande" (torneremo poi a discutere su questa scelta);
2. sia ω_n una radice n-esima primitiva di 1, e calcoliamo $\psi_i = \varphi(\omega^i)$ per $i = 0, \dots, n-1$ attraverso una IDFT;
3. calcoliamo le inverse delle matrici ψ_i;
4. interpoliamo una serie che assuma i valori ψ_i nelle radici n-esime di 1: $\psi(z) = \mathrm{DFT}(\psi_i)$.

Sappiamo dal teorema 3.1 che $\varphi^{-1}(z) \in \mathcal{W}$, quindi ha grado numerico finito: pertanto, facendo crescere del numero n dei punti di interpolazione riusciamo a ricostruirla esattamente. Tuttavia, il valore di n necessario non è noto a priori. Possiamo però adottare un algoritmo in cui cerchiamo di stimare n per passi successivi:

1. scegliamo come valore iniziale n uguale al grado numerico di U;
2. calcoliamo ψ come dall'algoritmo precedente;
3. l'approssimazione trovata è "soddisfacente"? Se non lo è, poniamo $n \leftarrow 2n$ e ricominciamo.

Come criterio di stop, nel nostro algoritmo (in cui ci interessa solo calcolare i coefficienti di grado $-N, \dots, N$ della φ^{-1}) abbiamo utilizzato

$$\max_{k=-N,\dots,N} \left\| \psi_k^{(2n)} - \psi_k^{(n)} \right\|_\infty < \varepsilon$$

scegliendo $\varepsilon = 10^{-11}$

Un'osservazione importante nell'implementazione è che per come è strutturata la DFT (l'algoritmo si basa su raddoppi successivi di dimensione), durante il calcolo delle DFT di lunghezza $2n$ riusciamo a riutilizzare i calcoli effettuati nei calcoli di dimensione n. Quindi nel calcolare la complessità computazionale di questo algoritmo possiamo considerare solo le DFT relative al valore di stop $\bar{n}$: otteniamo allora l'usuale $O(m^3\bar{n} + m^2\bar{n}\log\bar{n})$.

4.4 Calcolo del vettore stazionario

Abbiamo ora tutti i mezzi di cui abbiamo bisogno per calcolare velocemente il vettore stazionario. Il sistema da risolvere nel caso M/G/n è del tipo

$$\left[\begin{array}{c|ccc} \pi_0 & \pi_1 & \pi_2 & \dots \end{array}\right] \left[\begin{array}{c|ccc} B_0 & B_1 & B_2 & \dots \\ \hline B_{-1} & A_0 & A_1 & \dots \\ \vdots & A_{-1} & A_0 & \ddots \\ B_{-N} & \vdots & A_{-1} & \ddots \\ & A_{-n} & \ddots & \ddots \\ & & \ddots & \ddots \end{array}\right] = \left[\begin{array}{c|ccc} \pi_0 & \pi_1 & \pi_2 & \dots \end{array}\right].$$

Usiamo in seguito una notazione più compatta per i blocchi in cui abbiamo diviso la matrice, indicando l'equazione in questo modo:

$$\left[\begin{array}{c|c} \pi_0 & \pi_+ \end{array}\right] \left[\begin{array}{c|c} B_0 & B \\ \hline C & A \end{array}\right] = \left[\begin{array}{c|c} \pi_0 & \pi_+ \end{array}\right]$$

Possiamo risolvere il sistema effettuando questi passaggi:

$$\left[\begin{array}{c|c} \pi_0 & \pi_+ \end{array}\right] \left[\begin{array}{c|c} I - B_0 & -B \\ \hline -C & I - A \end{array}\right] = \left[\begin{array}{c|c} 0 & 0 \end{array}\right]$$

$$\begin{cases} \pi_0(I - B_0) - \pi_+ C = 0 \\ -\pi_0 B + \pi_+(I - A) = 0 \end{cases}$$

$$\begin{cases} \pi_0(I - B_0 - B(I - A)^{-1}C) = 0 \\ \pi_+ = \pi_0 B(I - A)^{-1} \end{cases} \tag{4.5}$$

Possiamo interpretare questi calcoli formali come operazioni su elementi di ℓ^1, in quanto le proprietà di stocasticità per il vettore π e per la matrice associata alla coda implicano banalmente che $\pi \in \ell^1(\mathbb{N})$ e che P è un operatore continuo di norma 1 su ℓ^1.

La prima equazione delle (4.5) è un semplice sistema lineare omogeneo $m \times m$ che possiamo risolvere con i metodi standard dell'analisi numerica (ad esempio l'eliminazione di Gauss); il punto critico in tutti i nostri passaggi è il calcolo di $(I - A)^{-1}$, cioè l'inversione di una matrice semi-infinita (vista come operatore su $\ell^1(\mathbb{N})$): in particolare dovremo calcolare $B(I - A)^{-1}$. A tal fine indichiamo con $\varphi(z) \in \mathcal{W}$ la serie dell'algebra di Wiener associata a $I - A$ (ossia, $I - A = T_\infty[\varphi]$) e utilizziamo questo algoritmo per il calcolo [2, 3]:

1. utilizziamo la tecnica di shift della sezione 4.1 per ottenere da φ una serie $\tilde{\varphi}(z)$ non singolare per $|z| = 1$;

2. applichiamo l'algoritmo della sezione 4.3 per calcolare i coefficienti centrali $H_{-N}, \ldots, H_N$ di $\tilde{\varphi}^{-1}(z)$;

3. utilizzando i coefficienti centrali di H appena determinati, calcoliamo la fattorizzazione canonica $\tilde{U}\tilde{L}$ di $\tilde{\varphi}$ come descritto nella sezione 4.2;

4. recuperiamo la fattorizzazione canonica debole $I - A = UL$ della $\varphi(z)$ con lo shift all'indietro di $\tilde{L}$ (come dalle equazioni (4.2));

5. tronchiamo le matrici U ed L alla più piccola potenza di 2 maggiore del grado numerico di $U(z)$, $L(z)$ e $B(z)$ (in modo da non avere errori numerici aggiuntivi dovuti al troncamento), quindi utilizziamo l'algoritmo della sezione 2.3 per calcolare U^{-1} e L^{-1};

6. si ha $B(I-A)^{-1} = B(L^{-1}U^{-1}) = (BL^{-1})U^{-1}$, che possiamo calcolare con due prodotti Toeplitz–vettore con l'algoritmo della sezione 2.2.

Tutte le operazioni coinvolte hanno un costo computazionale stimabile con $O(m^2 n \log n + m^3 n)$ operazioni, dove n è la dimensione a cui abbiamo scelto di troncare le serie di Wiener al punto 5.

L'algoritmo qui enunciato è stato descritto per la prima volta in letteratura in [3].

4.5 Esperimenti numerici

Come complemento alla trattazione teorica, abbiamo sviluppato un programma per il calcolo della fattorizzazione canonica come descritto nell'algoritmo appena esposto. Il programma è stato scritto in linguaggio C++ utilizzando la libreria per l'algebra lineare *TNT – Template Numerical Toolkit* (`http://math.nist.gov/tnt/`) e la routine per la DFT del pacchetto *ALP – Algèbre Linéaire pour les Polynômes* (`http://www-sop.inria.fr/-saga/Bernard.Mourrain/ALP/`).

I risultati ottenuti sono stati soddisfacenti: in particolare riportiamo i tempi relativi a due code utilizzate come esempio, la prima costruita appositamente, la seconda derivante da un modello nell'ambito delle telecomunicazioni [5]:

- $A(z) = (1 - 2\lambda)Cz^{-1} + \lambda Cz + \frac{\lambda}{m}Ez^2, \quad m = 10.$
  (positiva ricorrente per $\lambda < 0.2$)

λ	drift	steps	tempo(secondi)	errore
0.15	-0.25	256	0.052	7.70589e-14
0.19	-0.05	1024	0.131	3.98173e-13
0.195	-0.02	2048	0.242	3.15515e-13
0.199	-0.005	8192	1.017	1.54942e-12

- $A(z) = \frac{e^{-\lambda}}{mz^m} e^{\lambda z} \operatorname{Diag}(1, z, z^{m-1})E, \quad m = 15.$
  (positiva ricorrente per $\lambda < 8$)

λ	drift	steps	tempo(secondi)	errore
7.5	-0.5	2048	0.663	4.13803e-13
7.9	-0.1	8192	2.229	1.95295e-12
7.99	-0.03	32768	9.206	2.3238e-11

Qui E è la matrice $E_{ij} = 1$, C è la matrice di shift con $C_{i,i+1} = 1$, $C_{m,1} = 1$ e tutti gli altri elementi nulli, e l'errore è calcolato come il massimo della norma-1 delle matrici $m \times m$ $(I - A - UL)_i$. I tempi sono riferiti a un processore Intel Pentium 4 2.8 GHz; il programma è stato compilato con GNU gcc con le opportune opzioni di ottimizzazione attivate.

Dai dati si può verificare la dipendenza circa lineare del tempo di calcolo rispetto al drift della coda. Va però detto che il programma è stato sviluppato solo a titolo di verifica teorica e pertanto non è stata eseguita una ottimizzazione spinta a livelli più strettamente "informatici"; in modo particolare in un'implementazione orientata alle performance andrebbe modificata la parte della DFT ricorrendo a librerie più efficienti e robuste (per esempio la *FFTW – Fastest Fourier Transform in the West*, `http://www.fftw.org/`).

4.6 Conclusioni

L'algoritmo qui descritto risolve il problema richiesto in modo stabile e sufficientemente performante: l'ordine di complessità è pressoché ottimale e il guadagno si evidenzia soprattutto in code con drift molto vicino a zero, in cui le serie coinvolte hanno grado numerico particolarmente elevato.

Il modello trattato si adatta a tutte le code in cui la dipendenza dall'ambiente si può modellizzare con una catena di Markov; quindi gestisce situazioni come ad esempio l'aumento temporaneo del traffico causato da un incidente stradale, ma non è adattabile a modelli in cui non si ha omogeneità rispetto al tempo: ad esempio, supporre diverse condizioni di traffico in diversi momenti della giornata. In quest'ultimo caso infatti si perde la struttura Toeplitz della matrice coinvolta, che è quella che ha permesso di sviluppare un algoritmo veloce.

Bibliografia

[1] R. Bevilacqua, D. A. Bini, M. Capovani, O. Menchi. *Metodi Numerici.* Zanichelli, 1992.

[2] D. A. Bini, G. Latouche, B. Meini. *Numerical Methods for Structured Markov Chains.* Oxford University Press, 2005.

[3] D. A. Bini, B. Meini. *Non-Skip-Free M/G/1-Type Markov Chains and Laurent Matrix Power Series.* Linear Algebra Appl. 386:187-206, 2004.

[4] A. Böttcher, B. Silbermann. *Introduction to Large Truncated Toeplitz Matrices.* Universitext. Springer–Verlag, New York, 1999.

[5] H. R. Gail, S. L. Hantler, A. G. Konheim, B. A. Taylor. *An Analysis of a Class of Telecommunications Models.* Performance Evaluation, 21:151-161, 1994.

[6] H. R. Gail, S. L. Hantler, B. A. Taylor. *Spectral Analysis of M/G/1 and G/M/1 Type Markov Chains.* Adv. in Appl. Probab., 28(1):114-165, 1996.

[7] J. R. Norris. *Markov Chains.* Cambridge Series on Statistical and Probabilistic Mathematics. Cambridge University Press, 3rd edition, 1999.

[8] R. S. Varga. *Matrix Iterative Analysis.* Springer Series in Computational Mathematics, vol. 27. Springer–Verlag, Berlin, expanded edition, 2000.