Università degli Studi di Padova
Dipartimento di Scienze Statistiche
Corso di Laurea Triennale in

Statistica Economia e Finanza

RELAZIONE FINALE

UN ALGORITMO VELOCE PER LA DIFFERENZA FRAZIONARIA

Relatore Prof. Luisa Bisaglia
Dipartimento di Scienze Statistiche

Laureando: Matteo Boscato
Matricola N. 1052933

Anno Accademico 2014/2015

Indice

Introduzione

Questo lavoro prende spunto da un articolo di Andreas Noack Jensen e Morten Ørregaard Nielsen (2014), i quali propongono un algoritmo veloce per la differenza frazionaria di una serie storica. Grazie a questo algoritmo la velocità di calcolo, ovvero il numero di operazioni aritmetiche necessarie per differenziare una serie storica, si riduce sensibilmente rispetto all'algoritmo standard utilizzato per il medesimo fine. Nell'implementazione standard infatti, la velocità di calcolo è dell'ordine di T^2, dove T è la lunghezza della serie. Al contrario invece, il nostro algoritmo permette una velocità di calcolo dell'ordine di $TlogT$. Una differenza sostanziale, in particolar modo per campioni di medio-grandi dimensioni. L'algoritmo proposto è basato su convoluzioni circolari ed è disegnato per sfruttare al meglio implementazioni molto efficienti della trasformata discreta di Fourier, come la trasformata veloce di Fourier (Cooley e Tukey, 1965). In questo elaborato andremo inizialmente a presentare e definire i modelli *ARFIMA(p,d,q)*, modelli sui quali applicheremo l'algoritmo proposto, per poi confrontarlo con l'algoritmo standard utilizzato. Introdurremo poi il dominio delle frequenze e definiremo la trasformata discreta di Fourier e la trasformata veloce di Fourier, poiché l'algoritmo proposto si basa su esse. Andremo poi quindi a presentare dettagliatamente l'algoritmo veloce per la differenza frazionaria di una serie storica, mostrando come esso riesca, grazie ad alcuni importanti teoremi, a calcolare la differenza frazionaria basandosi sulla trasformata discreta di Fourier. Confronteremo a questo punto il nuovo algoritmo con l'algoritmo standard comunemente utilizzato, per capire se effettivamente la velocità di calcolo per la differenza frazionaria sia sostanzialmente inferiore. Eseguiremo, quindi uno studio di simulazione, utilizzando il software R, per campioni di serie storiche di grandezza da *100* a *100.000*. I risultati ottenuti verranno poi riportati graficamente, al fine di rendere visibile la notevole differenza di velocità di calcolo dei due algoritmi.
A questo punto andremo a verificare se i due algoritmi utilizzati abbiano differenziato in maniera adeguata o meno le serie. Per fare ciò ci avvarremo di due diversi metodi di stima del parametro di differenziazione *d*, e calcoleremo alcune statistiche, necessarie al

fine di verificare la bontà del calcolo svolto dai due algoritmi. Infine, riporteremo i risultati ottenuti dallo studio di simulazione e dalla verifica dei due algoritmi, traendo delle conclusioni per sostenere la tesi che l'algoritmo proposto sia una valida (o migliore) alternativa all'algoritmo standard utilizzato per il calcolo della differenza frazionaria.

Capitolo 1

Modelli ARFIMA(p,d,q) e dominio delle frequenze

1.1 I modelli ARFIMA(p,d,q)

I modelli *ARFIMA(p,d,q)* sono stati introdotti da Hosking (1981) e da Granger e Joyeux (1980). Questi modelli, detti anche *ARIMA(p,d,q) frazionari*, vennero introdotti come una generalizzazione degli usuali modelli *ARIMA(p,d,q)*, i quali possono essere non stazionari quando $d = 1$. Il parametro di differenziazione tuttavia può assumere anche valori intermedi tra $d = 0$ e $d = 1$, ed è per questo motivo che sono stati introdotti i modelli *ARFIMA*. In presenza, quindi, di processi non stazionari è necessario trasformare la serie per ottenerne una stazionaria, a cui sia possibile applicare i modelli *ARMA(p,q)*. Per fare ciò possiamo ad esempio pensare di differenziare la serie, nel momento in cui quest'ultima sia non stazionaria in media. Se quindi $X_t = (1 - B)^d Y_t$, con $d \in N$, diremo che Y_t è un processo integrato di ordine d e viene indicato come *ARIMA(p,d,q)*, dove d viene chiamato parametro di differenziazione o integrazione. L'equazione corrispondente a questo processo è dunque la seguente:

$$\Phi(B)(1 - B)^d X_t = \Theta(B)\varepsilon_t \qquad (1.1.1)$$

dove ε_t~*WN(0,*σ_ε^2*)* e *Φ(·)* e *Θ(·)* sono due polinomi di grado rispettivamente *p* e *q*[1].
Si noti che i modelli *ARMA(p,q)* possono essere espressi come *ARIMA(p,0,q)*, dove il parametro di differenziazione *d* è posto uguale a *0*, poiché non è necessario differenziare il processo. Se il parametro $d \geq 1$, allora la serie originaria X_t non è stazionaria. Per ottenere un processo stazionario, X_t deve essere differenziata *d* volte.
Come abbiamo detto poco fa i modelli *ARFIMA(p,d,q)* generalizzano i modelli *ARIMA(p,d,q)* poiché il parametro di differenziazione *d* può assumere un qualsiasi valore reale ($d \in R$). In questo modo si può cosi generalizzare l'equazione (1.1.1). Inizialmente osserviamo che se *d* è intero, allora:

$$(1-B)^d = \sum_{k=0}^{d} \binom{d}{k} (-1)^k B^k$$

dove il coefficiente binomiale è uguale a:

$$\binom{d}{k} = \frac{d!}{k!\,(d-k)!} = \frac{\Gamma(d+1)}{\Gamma(k+1)\Gamma(d-k+1)}$$

e *Γ(·)* è la funzione Gamma. Dato che la funzione Gamma è definita per tutti i valori reali, possiamo estendere la precedente definizione del coefficiente binomiale al caso $d \in R$:

$$(1-B)^d = \sum_{k=0}^{d} \binom{d}{k} (-1)^k B^k = \sum_{k=0}^{\infty} \pi_k B^k \qquad (1.1.2)$$

dove:

---

[1] I due polinomi $\Phi(B)$ e $\Theta(\cdot)$ sono tali per cui $\Phi(B) = 1 - \Phi_1 B - \Phi_2 B^2 - \ldots - \Phi_p B^p$, $\Theta(B) = \Theta_1 B + \Theta_2 B^2 + \ldots + \Theta_p B^p$, mentre l'operatore ritardo B è tale che $B^j X_t = X_{t-j}$ per $j = \ldots, -1, 0, 1, \ldots$

$$\pi_k = \frac{\Gamma(k-d)}{\Gamma(k+1)\Gamma(-d)} = \prod_{0<j\leq k} \frac{j-1-d}{j}, \qquad j = 0,1,2, \ldots.$$

Diamo dunque ora la definizione di processo *ARFIMA(p,d,q)*:

Definizione 1.1.1 *X = (X_t , t ∈ Z)* è un processo *ARFIMA(p,d,q)* con *d ∈ (-1/2 , 1/2)*, se è stazionario e se soddisfa l'equazione alle differenze:

$$\Phi(B)(1-B)^d X_t = \Theta(B)\epsilon_t \qquad (1.1.3)$$

dove ε_t~WN(0,σ_ε^2) e *Φ(·)* e *Θ(·)* sono due polinomi di grado *p* e *q* rispettivamente.

Quando *d ∈ (0 , 1/2)* il processo *ARFIMA(p,d,q)* è un processo stazionario a memoria lunga. Se poniamo invece il parametro di differenziazione *d = 0* riotteniamo il solito modello *ARMA(p,q)*. Infine, se poniamo i parametri *p = q = 0* il processo *X = (X_t , t ∈ Z)* è chiamato rumore integrato frazionario, ed è denotato *ARFIMA(0,d,0)*.

1.2 Dominio delle frequenze

La nozione che una serie storica presenti un comportamento ripetitivo o regolare con il passare del tempo è di fondamentale importanza perché distingue l'analisi delle serie storiche dalla statistica classica. Possiamo infatti affermare come il concetto di regolarità di una serie possa essere espresso nel miglior modo in termini di variazioni periodiche del fenomeno sottostante che produce le serie, espresse da frequenze che sono regolate da seni e coseni. Da un punto di vista regressivo, l'approccio nel dominio del tempo può essere pensato come una regressione del presente sul passato, che esamina come una serie storica si evolve nel tempo, con strumenti come la funzione di autocorrelazione. L'approccio nel dominio delle frequenze, invece, può essere considerato come una

regressione del presente su seni e coseni periodici, poiché studia come queste componenti periodiche a diversa frequenza descrivono l'evoluzione di una serie storica. Ovviamente, la giustificazione primaria per utilizzare modelli alternativi a quelli utilizzati nel dominio del tempo deve stare nel suo potenziale per spiegare il comportamento di alcuni fenomeni empirici. In questo senso, una spiegazione che implica solo pochi tipi di oscillazioni primarie diventa più semplice e più significativa di una collezione di parametri stimati per alcune equazioni scelte. E' la tendenza dei dati osservati a mostrare fluttuazioni di tipo periodico che giustifica l'uso dell'approccio nel dominio delle frequenze. Tale approccio tratta la correlazione generata in serie storiche stazionarie che inizia trasformando le serie nel dominio delle frequenze. Questa semplice trasformazione lineare abbina essenzialmente i seni e i coseni di varie frequenze con i dati sottostanti. Un'altra ragione per applicare questo approccio è la convenienza statistica risultante dalle componenti periodiche essendo approssimativamente incorrelate. Questa proprietà facilita lo scrivere verosimiglianze basate su metodi statistici classici.

Entriamo dunque nel campo del dominio delle frequenze, ovvero in un contesto di studio di funzioni matematiche e segnali che vengono descritte mediante l'analisi dello spettro delle frequenze che lo costituiscono; invece di essere quindi descritte mediante il loro andamento nel tempo (dominio del tempo).

Se osserviamo un grafico di una funzione o di un segnale nel dominio del tempo, esso ci mostra come il segnale cambia nel tempo, mentre se osserviamo un grafico nel dominio delle frequenze ci mostra come e quanto un segnale si suddivide o è distribuito nelle varie bande di frequenza, definite all'interno di un dato range.

Un segnale o una funzione possono essere convertiti dalla rappresentazione nel dominio del tempo a quella nel dominio delle frequenze attraverso l'uso di operatori matematici chiamati trasformate, come la trasformata di Fourier; quest'ultima scompone un segnale nella sommatoria di un numero finito o infinito di onde sinusoidali. Lo spettro di tutte le frequenze componenti costituisce la rappresentazione del segnale nel dominio delle frequenze. Di solito, per visualizzare un segnale nel dominio delle frequenze si fa ricorso ad uno strumento chiamato periodogramma, di cui dopo parleremo.

La generale nozione di periodicità può essere fatta più precisamente introducendo alcune terminologie. Al fine di definire il tasso al quale una serie oscilla, diamo innanzitutto una definizione di ciclo, il quale è un periodo completo di una funzione seno o coseno definito su un intervallo di tempo unitario. Consideriamo il processo periodico:

$$X_t = Acos(2\pi wt + \text{ø}) \qquad (1.2.1)$$

per *t = 0, ± 1, ± 2, ...,* dove *w* è la frequenza, definito in cicli per unità di tempo con *A* che determina l' ampiezza della funzione e ø, chiamata fase, che determina il punto di origine della funzione coseno.

Possiamo usare un'identità trigonometrica[2] e scrivere (1.2.1) come:

$$X_t = U_1 cos(2\pi wt) + U_2 sen(2\pi wt) \qquad (1.2.2)$$

dove $U_1 = Acos\,\text{ø}$ e $U_2 = -\text{Asen}\,\text{ø}$ sono spesso utilizzati perché sono variabili casuali normali. L'ampiezza è $A = \sqrt{U_1^2 + U_2^2}$ e la fase è $\text{ø} = tan^{-1}(-U_2 / U_1)$.

Il processo casuale trattato è anche una funzione delle sue frequenze, definito dal parametro *w*. La frequenza è misurata in cicli per unità di tempo. Per *w=1*, la serie compie un ciclo per unità di tempo; per *w=0.5*, la serie compie un ciclo ogni due unità di tempo; per *w=0.25*, ogni quattro unità, e così via.

Consideriamo una generalizzazione di (1.2.2) che permette sovrapposizioni di serie periodiche con frequenze e ampiezze multiple:

$$x_t = \sum_{k=1}^{q} [U_{k1}\, cos(2\pi w_k t) + U_{k2}\, sen(2\pi w_k t)], \qquad (1.2.3)$$

dove U_{k1}, U_{k2}, per *k = 1, 2, ..., q*, sono variabili casuali indipendenti con media zero, con varianza σ_k^2 e le w_k sono frequenze distinte. Notiamo che (1.2.3) mostra il processo come una somma di componenti indipendenti, con varianza σ_k^2 per frequenze w_k. Usando l'indipendenza di U_s e l'indipendenza trigonometrica, è facile mostrare che la

---

[2] $\cos(\alpha \pm \beta) = \cos(\alpha)\cos(\beta) \mp sen(\alpha)sen(\beta)$.

funzione di autocovarianza del processo è:

$$\gamma(h) = \sum_{k=1}^{q} \sigma_k^2 \cos(2\pi w_k h), \qquad (1.2.4)$$

e notiamo che la funzione di autocovarianza è la somma di componenti periodiche con pesi proporzionali alle varianze σ_k^2. Dunque, x_t è un processo stazionario a media zero con varianza:

$$\gamma(\mathrm{h}) = E(x_t^2) = \sum_{k=1}^{q} \sigma_k^2$$

che mostra la varianza complessiva come una somma di varianze di ogni parte della componente.

Il sistematico ordinamento delle componenti delle frequenze fondamentali in una serie storica, costituisce uno dei principali obiettivi dell'analisi spettrale.

Se non conosciamo *w*, potremmo provare a stimare il modello usando una regressione non lineare con *w* come parametro. Un altro metodo è provare diversi valori di *w* in maniera sistematica. Il punto, comunque, è che se i dati contengono alcuni comportamenti ciclici noi siamo in grado di coglierli eseguendo queste regressioni.

Nel caso in cui provassimo a stimare il modello attraverso una regressione ponendo *w* come variabile, otterremo che i coefficienti di regressione $\widehat{U}_1(j/n)$ e $\widehat{U}_2(j/n)$, per ogni *j*, stanno essenzialmente misurando la correlazione dei dati con un'oscillazione sinusoidale a *j* cicli in *n* istanti di tempo. Dunque, un' appropriata misura della presenza di una frequenza di oscillazione di j cicli in n istanti di tempo nei dati sarebbe:

$$P(j/n) = \widehat{U}_1^2(j/n) + \widehat{U}_2^2(j/n), \qquad (1.2.6)$$

che è semplicemente una misura della correlazione quadratica. La quantità (1.2.6) è a volte chiamata periodogramma, ma noi chiameremo *P(j/n)* periodogramma scalato.

Infine, diciamo che non è necessario eseguire una regressione:

$$x_t = \sum_{j=0}^{n/2} U_1(j/n)\, cos(2\pi tj/n) + \; U_2\left(\frac{j}{n}\right) sen(2\pi tj/n) \qquad (1.2.7)$$

per ottenere i valori di U_1 e U_2 perché possono essere calcolati velocemente se *n* è un numero intero. Non vi sono errori in (1.2.7) perché ci sono *n* osservazioni e *n* parametri; la regressione stimata sarà perfetta. La trasformata discreta di Fourier(DTF) è una media pesata di valori complessi dei dati data da:

$$d(j/n) = \; n^{-1/2} \sum_{t=1}^{n} x_t \, exp(-2\pi itj/n)$$

$$= \; n^{-\frac{1}{2}} (\sum_{t=1}^{n} x_t \cos(-2\pi itj/n) - \; i \sum_{t=1}^{n} x_t \, sen(2\pi tj/n)) \qquad (1.2.8)$$

dove le frequenze *j/n* sono chiamate frequenze di Fourier o frequenze fondamentali. A causa del grande numero di ridondanze nel calcolo, la (1.2.8) potrebbe essere calcolata velocemente usando la trasformata veloce di Fourier(FFT). Notiamo che:

$$|d(j/n)|^2 = \; \frac{1}{n} \; (\sum_{t=1}^{n} x_t \cos\,(2\pi tj/n))^2 + \; \frac{1}{n} (\sum_{t=1}^{n} x_t \text{sen}(2\pi tj/n))^2 \qquad (1.2.9)$$

ed è questa quantità che è chiamata periodogramma, che ci consentirà di scoprire le componenti periodiche di una serie storica; scriveremo:

$$I(j/n) \; = \; |d(j/n)|^2.$$

Noi potremmo calcolare il periodogramma scalato (1.2.6), usando il periodogramma come:

$$P(j/n) = \; \frac{4}{n} \; I(j/n). \qquad (1.2.10)$$

1.3 Trasformata discreta di Fourier (DTF) e trasformata veloce di Fourier (FFT)

Abbiamo visto come le trasformate discrete di Fourier siano di fondamentale importanza nel calcolo di quantità come il periodogramma, utile per scoprire le componenti periodiche in una serie storica.

Per quando riguarda la trasformata discreta di Fourier (DTF), si tratta di un procedimento che converte una collezione finita di campioni equispaziati di una funzione in una collezione di coefficienti di una combinazione lineari di sinusoidi complesse, che vengono ordinate al crescere della frequenza. Si tratta, in sintesi, di uno strumento per rappresentare una funzione, la cui variabile può essere ad esempio il tempo, nel dominio delle frequenze. Le frequenze delle sinusoidi della combinazione lineare prodotta dalla trasformata sono multipli interi di una frequenza fondamentale, il cui periodo è la lunghezza dell'intervallo di campionamento. La trasformata discreta di Fourier richiede in ingresso una funzione discreta, i cui valori siano generalmente complessi, non nulli ed abbiano una durata limitata. Ciò rende la DTF molto utile per analizzare le frequenze contenute in un segnale. Per calcolarla, poi, in maniera efficiente si possono usare gli algoritmi per la trasformata veloce di Fourier.

La trasformata veloce di Fourier (FFT) è, infatti, un algoritmo ottimizzato per calcolare proprio la trasformata discreta di Fourier e la sua inversa. Questo algoritmo ci permette di ridurre notevolmente il numero di operazioni richieste per calcolare determinate quantità, come ad esempio il periodogramma. Tale algoritmo sarà di fondamentale importanza nella presentazione del nuovo metodo proposto per la differenza frazionaria, che introdurremo ora.

Capitolo 2

Un algoritmo veloce per la differenza frazionaria

2.1 Introduzione

Nelle simulazioni di serie storiche da modelli frazionalmente integrati, il costo computazionale è quasi esclusivamente associato al calcolo della differenza frazionaria. Infatti, il costo computazionale di questi calcoli può essere così grande che le stime o le simulazioni di serie storiche integrate frazionarie sono irrealizzabili quando la dimensione del campione è molto grande.

Deriveremo, dunque, un algoritmo per il calcolo delle differenze frazionarie basato su delle convoluzioni circolari. Il vantaggio di questo algoritmo è che è disegnato per sfruttare implementazioni molto efficienti della trasformata discreta di Fourier, ovvero la trasformata veloce di Fourier, che abbiamo precedentemente presentato. Il numero di operazioni aritmetiche richieste, e dunque la velocità di calcolo, dell'implementazione standard dell'operazione di differenza frazionaria è dell'ordine di T^2. L'algoritmo proposto è in grado di raggiungere l'ordine di $TlogT$. Per campioni di grandi dimensioni, la differenza di tempo computazionale è sostanziale. Per esempio, supponiamo di osservare un campione di dimensione $T = 100.000$, che è abbastanza ragionevole per applicazioni, ad esempio, in finanza. La differenza nel tempo di stima per l'implementazione standard contro la nostra implementazione dell'algoritmo alternativo della differenza frazionaria potrebbe essere dell'ordine di *4.5 minuti* contro *2 secondi*. Il costo computazionale con implementazioni standard sembra proibitivo nel caso si vogliano applicare tecniche bootstrap o si vogliano simulare campioni di grandi

dimensioni. Dall'altro lato, queste procedure rimangono fattibili con la nuova implementazione dell'operatore differenza frazionaria.

2.2 Algoritmo veloce per la differenza frazionaria

Consideriamo la serie storica X_t, che è osservata per *t = 1,.....,T*. Supponiamo di voler calcolare la differenza frazionaria:

$$Y_t = (1-B)_+^d X_t = \sum_{j=0}^{t-1} \pi_j(-d) X_{t-j}\,, \qquad t = 1, \ldots, T \qquad (2.2.1)$$

dove i coefficienti frazionari $\pi_j(u)$ sono definiti come i coefficienti in un'espansione di $(1 - z)^{-u}$, che sono:

$$\pi_j(u) = \frac{u(u+1)\cdots(u+j-1)}{j!}, \qquad j = 0,1, \ldots \qquad (2.2.2)$$

Notiamo che la sommatoria in (2.2.1) è troncata a *t – 1* perché noi osserviamo X_t iniziando al tempo *t = 1*. La sottoscritta '+' sull'operatore differenza frazionaria indica dunque che solo le osservazioni su X_t con un indice di tempo positivo sono incluse nella sommatoria. Se avessimo osservazioni prestabilite (valori iniziali) su X_t che vogliamo includere, allora la sommatoria sarebbe estesa per includerli al meglio.
Il calcolo standard della differenza frazionaria in (2.2.1) è rappresentato come una convoluzione circolare di due serie $X = (X_t)_{t=1}^T$ e $q = (\pi_{t-1}(-d))_{t=1}^T$. Quindi, la serie temporale $Y = (Y_t)_{t=1}^T$ con *t-esimo* elemento dato nella (2.2.1) può essere scritta come:

$$Y_t = \sum_{j=1}^{t} q_j X_{t-j+1}, \qquad t = 1, \ldots, T. \qquad (2.2.3)$$

Poiché il numero di operazioni aritmetiche richieste in ogni somma nella (2.2.3) è di ordine *t*, l'intera operazione di convoluzione lineare per *t = 1,.....,T* è dell'ordine T^2.

Il nuovo algoritmo per l'operatore differenza frazionaria prende vantaggio da una trasformazione nel dominio delle frequenze, e definiamo dunque la trasformata discreta di Fourier $f = (f_j)_{j=1}^{T}$ di una serie $a = (a_t)_{t=1}^{T}$ come la soluzione dell'equazione $a = T^{-1}Ff$, dove F è la matrice di Fourier con *(j,k)-esimo* elemento $(F)_{jk} = w_t^{(j-1)(k-1)}$ e $w_t = e^{i2\pi/T}$ con $i = \sqrt{-1}$ che denota l'unità immaginaria. Ogni elemento di a può quindi essere espresso in termini dei coefficienti di Fourier f_j e potenze di w_t come:

$$a_t = \frac{1}{T}\sum_{j=1}^{T} f_j w_t^{(t-1)(j-1)}, \qquad t = 1, \ldots, T \tag{2.2.4}$$

Poiché F è simmetrica e $F\bar{F} = TI_T$, dove la barra denota la coniugata complessa, l'operazione inversa è $(T^{-1}F)^{-1} = \bar{F}$, cioè la coniugata complessa di ogni elemento in F, così che$f_j = \sum_{t=1}^{T} a_t w^{-(t-1)(j-1)}$. Quindi, la matrice $\bar{F}$ rappresenta la trasformata discreta di Fourier, mentre $T^{-1}F$ è la trasformata inversa.

La convoluzione circolare di due serie a e b di lunghezza T è denotata $a \circledast b$ e definita come:

$$(a \circledast b)_t = \sum_{j=1}^{T} a_j b_{t-j+1}, \qquad t = 1, \ldots, T \tag{2.2.5}$$

dove le sequenze sono estese periodicamente così che $a_{j+nT} = a_j$ e $b_{j+nT} = b_j$ per ogni $n = 0, \pm 1, \pm 2, \ldots$

Nel Teorema 1 affermiamo la versione finita del teorema della convoluzione circolare, il quale mostra come la convoluzione circolare delle sequenze finite nella (2.2.5) possa essere calcolata applicando la trasformata discreta di Fourier.

Teorema 1. Siano $a = (a_t)_{t=1}^{T}$ e $b = (b_t)_{t=1}^{T}$ due sequenze. Allora:

$$a \circledast b = T^{-1}F(\bar{F}a \circ \bar{F}b), \tag{2.2.6}$$

dove il simbolo '$\circ$' denota la moltiplicazione elemento per elemento.

Dimostrazione.

Siano $f = \bar{F}a$ e $g = \bar{F}b$ le trasformate discrete di Fourier di a e b rispettivamente. Bisogna quindi mostrare che $a \circledast b = T^{-1}F(f \circ g)$. Per fare ciò, inseriamo le espressioni per *a* e *b* in termini delle loro trasformate discrete di Fourier,

$$(a \circledast b)_t = \sum_{j=1}^{T} a_j \, b_{t-j+1} = \sum_{j=1}^{T} \left(T^{-1} \sum_{s=1}^{T} f_s w_T^{(j-1)(s-1)} \right) (T^{-1} \sum_{u=1}^{T} g_u \, w_T^{(t-j)(u-1)})$$

$$= T^{-2} \sum_{s=1}^{T} \sum_{u=1}^{T} f_s g_u \sum_{j=1}^{T} w_T^{(t-1)(s-1)+(j-1)(s-u)}$$

$$= T^{-2} \sum_{s=1}^{T} \sum_{u=1}^{T} f_s g_u w_T^{(t-1)(s-1)} \sum_{j=1}^{T} w_t^{(j-1)(s-u)}$$

$$= T^{-1} \sum_{s=1}^{T} f_s g_s w_T^{(t-1)(s-1)}$$

dove l'ultima uguaglianza segue a causa del ben noto risultato:

$$\sum_{j=1}^{T} w_T^{(j-1)k} = \begin{cases} T \; if \; k = 0 \\ 0 \; if \; k \neq 0 \end{cases}$$

Questo mostra che i coefficienti di Fourier di $a \circledast b$ sono dati dal prodotto elemento per elemento di coefficienti di Fourier di *a* e *b*.

■

Il prossimo teorema presenta il risultato principale, che è mostrare come il teorema di convoluzione circolare possa essere usato per calcolare la differenza frazionaria nella (2.2.1), o equivalentemente nella (2.2.3), con la trasformata discreta di Fourier. Per un vettore *T*1* chiamato *a*, sia

$$\tilde{a} = [a'_T, 0'_{T-1}]' \qquad (2.2.7)$$

il vettore *(2T-1)*1* che consiste in un'estensione di a con *T-1 zeri*.

Teorema 2. La serie storica con differenza frazionaria in *Y* nella (2.2.3) può essere calcolata come i primi *T* elementi del vettore *(2T-1)*1*:

$$T^{-1}F(\bar{F}\tilde{q}\circ\bar{F}\tilde{X}) \tag{2.2.8}$$

Dimostrazione.

Dall'equazione (2.2.6), si ottiene che il *t-esimo* elemento della (2.2.8) è uguale a $(\tilde{q} \circledast \tilde{X})_t$. Per di più, per t=1,...T, abbiamo dalla definizione (2.2.5) che:

$$\left(\tilde{q} \circledast \tilde{X}\right)_t = \sum_{j=1}^{2T-1} \tilde{q}_j \tilde{X}_{t-j+1}$$

$$= \sum_{j=1}^{t} \tilde{q}_j \tilde{X}_{t-j+1} + \sum_{j=t+1}^{2T-1} \tilde{q}_j \tilde{X}_{2T+t-j}$$

$$= \sum_{j=0}^{t-1} \pi_j(-d) X_{t-j}$$

Perché $\tilde{q}_j = \pi_{j-1}(-d)$ per $j = 1, \ldots, T$ e $\tilde{q}_j = 0$ per $j \geq T+1$, mentre $\tilde{X}_{t-j+1} = X_{j+1}$ per $j = 1, \ldots, t$ e $X_{2T+t-j} = 0$ per $t+1 \leq j \leq T$ dalla definizione (2.2.7).

■

Il significato del teorema 2 sta nel fatto che la trasformata discreta di Fourier può essere calcolata molto efficacemente con l'utilizzo dell'algoritmo della trasformata veloce di Fourier, dove il numero di operazioni aritmetiche richieste è proporzionale a *TlogT*. Poiché l'operazione nella (2.2.8) applica solo tre trasformate discrete di Fourier e una moltiplicazione elemento per elemento di due vettori (che è di ordine *T*), l'algoritmo differenza frazionaria (2.2.8) nel teorema 2 è esso stesso di ordine *TlogT*.

Da notare che il nostro risultato nel teorema 2 fornisce un calcolo esatto della differenza frazionaria nella (2.2.1) e che nessuna approssimazione è coinvolta. Infine, è anche da notare che, a seconda della particolare implementazione della trasformazione di Fourier applicata , potrebbe essere necessario estendere le serie $\tilde{X}$ e $\tilde{q}$ ad una lunghezza più grande rispetto alla *2T-1* usata nella (2.2.7).

In modo specifico, alcune implementazioni della trasformata veloce di Fourier richiedono che la lunghezza sia una potenza di due, e in quel caso, $\tilde{X}$ e $\tilde{q}$ dovrebbero essere estese con degli zero fino ad una lunghezza uguale alla più piccola potenza di due che è almeno di *2T-1*.

Capitolo 3

Esperimento Monte Carlo

3.1 Presentazione dell'esperimento

Tramite un'esperimento di Monte Carlo abbiamo comparato l'algoritmo standard *'diffseries',* proposto nel pacchetto *fracdiff* di R con l'algoritmo *'fracdiff'*, al fine di capire se veramente tale algoritmo ci permette di calcolare la differenza frazionaria di una serie storica in un tempo computazionale sostanzialmente ridotto rispetto all'algoritmo standard. Il metodo Monte Carlo si basa sulla ripetizione di simulazioni casuali per ottenere risultati numerici, nel nostro caso, i tempi impiegati dai due algoritmi per l'operazione di differenza frazionaria.

Per realizzare l'esperimento di Monte Carlo abbiamo generato i dati a partire da un modello *ARFIMA(0,d,0),* grazie alla funzione *fracdiff.sim* presente nel pacchetto R chiamato *fracdiff*. Abbiamo stabilito alcuni diversi valori attribuibili al parametro di differenziazione *d* e, per ogni valore di *d* assegnato, abbiamo ripetuto l'esperimento, simulando dei campioni di serie storiche di dimensioni da *100* a *100.000*. Poiché siamo interessati solamente a processi a memoria lunga in senso stretto, abbiamo considerato valori di *d* appartenenti all'intervallo $(0, 1/2)$, e, in particolare abbiamo considerato $d = 0.1, 0.3, 0.45, 0.49$.

Per ogni valore di d assegnato abbiamo generato delle serie storiche di dimensioni da *100* fino a *100.000*, con differenza di *100* valori tra una serie e l'altra. Dopo aver generato queste serie, abbiamo applicato i due diversi algoritmi per la differenza frazionaria. Abbiamo inizialmente applicato l'algoritmo standard, grazie alla funzione *'diffseries'* presente sempre nel pacchetto R *fracdiff*. Abbiamo poi applicato l'algoritmo

'*fracdiff*', proposto nel capitolo precedente, alle stesse serie sfruttando l'applicazione della trasformata veloce di Fourier; prima di applicare questo algoritmo abbiamo implementato in R la routine *fracdiff*, presente nell'articolo da cui trae spunto questo studio. Grazie ad una funzione chiamata *microbenchmark*, presente nella *library(microbenchmark)* siamo riusciti a calcolare in maniera dettagliata i tempi impiegati dai due diversi algoritmi per differenziare la serie. Dopo aver ottenuto i tempi per ogni serie storica differenziata con i due diversi algoritmi, abbiamo creato una matrice dove li abbiamo riposti: nella prima colonna abbiamo collocato la numerosità campionaria delle varie serie, nella seconda e terza colonna invece, i tempi impiegati rispettivamente dall'algoritmo standard e dal nuovo algoritmo proposto.

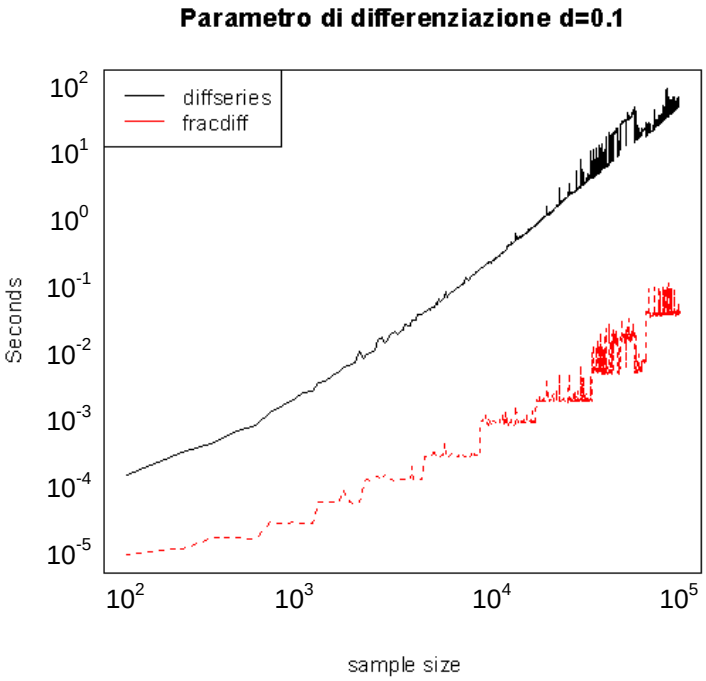
Fatto ciò, abbiamo rappresentato graficamente i risultati ottenuti. I tempi computazionali risultanti sono rappresentati con assi logaritmici, in modo tale da mostrare la differenza degli ordini degli algoritmi. E' stato realizzato un grafico per ogni studio di simulazione con differente valore assegnato al parametro *d*, per verificare se la scelta di un diverso valore del parametro possa in qualche modo alterare o modificare l'esito dell'esperimento.

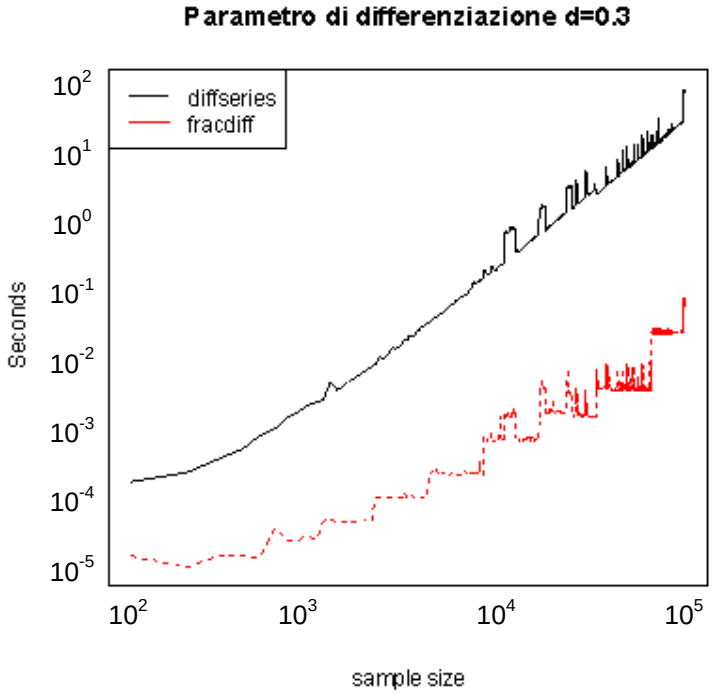
3.2 Risultati

Illustriamo numericamente la differenza nel costo computazionale tra l'implementazione standard nella (2.2.1) o nella (2.2.3) e l'algoritmo (2.2.8) nel teorema 2 per una gamma di campioni di dimensioni T. L'algoritmo basilare calcola direttamente la convoluzione lineare, dove il numero di operazioni aritmetiche richieste è dell'ordine di T^2. Il tempo computazionale ci si aspetta sia proporzionale al numero di operazioni aritmetiche, e conseguentemente dell'ordine di T^2. Per quanto riguarda l'algoritmo basato sulla trasformata veloce di Fourier ci aspettiamo, come abbiamo già ribadito in precedenza, che necessiti di un numero di operazioni che richieda un costo computazionale di ordine $TlogT$.

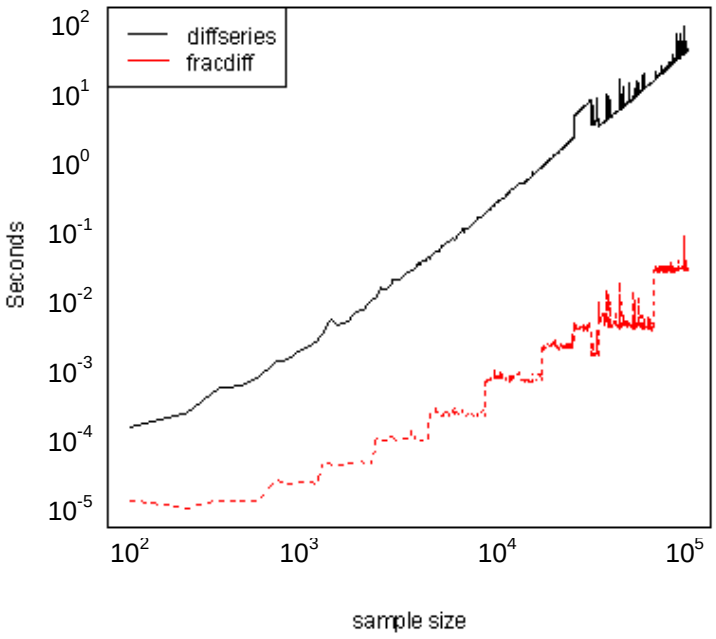
Osserviamo dunque ora i grafici ottenuti con il comando di R *matplot*, dopo aver differenziato i nostri campioni con i due diversi algoritmi. Sull'asse delle ascisse

troveremo la numerosità campionaria delle varie serie, mentre sull'asse delle ascisse i tempi impiegati dai due algoritmi. Avremo un grafico per ogni diverso valore assegnato al parametro di differenziazione *d*:

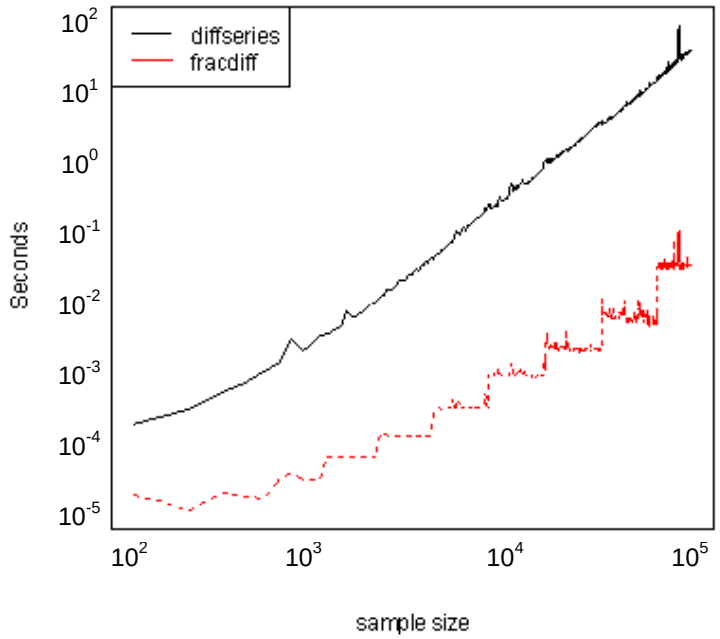




Parametro di differenziazione d=0.45



Parametro di differenziazione d=0.49



Come avevamo previsto nel presentare il nuovo algoritmo, l'ordine di grandezza del tempo richiesto dai due algoritmi per differenziare tutte le serie è sostanzialmente diverso. Possiamo ben vedere dai grafici infatti come per il nuovo algoritmo, il tempo venga rappresentato da una funzione che presenta dei salti a ogni potenza di due, come è previsto dalla FFT; al contrario, il grafico dell'algoritmo standard cresce in maniera quadratica.

Poiché hanno scale differenti, il nostro metodo diventa rapidamente molto più veloce al crescere della dimensione del campione. Comparato con l'algoritmo standard, l'algoritmo *'fracdiff'* è quasi 1620 volte più veloce in R. Ovviamente la ragione più importante per avere un algoritmo veloce per la differenza frazionaria sono le sue ripetute applicazioni, per esempio, nelle stime, simulazioni, o procedure bootstrap.

Nel nostro caso, dunque, riduce sostanzialmente il costo computazionale per la differenza frazionaria.

3.3 Analisi dei risultati ottenuti

Dopo aver differenziato il campione di serie storiche con i due diversi algoritmi, quali *'diffseries'* e *'fracdiff'* ci siamo concentrati sulla stima del parametro di differenziazione *d*, utilizzando due diversi metodi di stima: il metodo *GPH* (Geweke e Porter-Hudak, 1983) e quello di *Whittle* (Whittle,1951). Siamo interessati, infatti, alla stima di tale parametro poiché vogliamo stabilire se i due algoritmi abbiano effettivamente differenziato correttamente le serie storiche simulate.

Per condurre tale esperimento abbiamo generato *1000* serie storiche di lunghezza *100, 250, 500, 750, 1000,* ripetendo ogni volta tale procedimento per i diversi valori assegnati al parametro d che, come in precedenza, sono *0.1, 0.3, 0.45, 0.49,* tutti all'interno della soglia di non stazionarietà.

Presentiamo brevemente ora i due metodi di stima utilizzati. Per quanto riguarda il metodo di stima *GPH*[3], quest'ultimo stima il parametro frazionario d nei modelli *ARFIMA(p,d,q)* con il metodo di Geweke e Porter-Hudak (GPH). Lo stimatore *GPH* è basato su un'equazione di regressione che utilizza la funzione periodogramma come una stima della densità spettrale.

Lo *stimatore di Whittle*, invece, si basa su un'approssimazione della funzione di verosimiglianza. Vi sono diversi approcci, nella letteratura delle serie storiche, che permettono di determinare con una buona approssimazione tale funzione. Questi approcci si basano proprio sul lavoro svolto da Whittle (1951), il quale ha introdotto quest'approssimazione nel caso di variabili casuali debolmente indipendenti. Per utilizzare questo stimatore abbiamo utilizzato una particolare routine, al fine di applicare tale metodo di stima con il software R.

Dunque, nello specifico, ad ogni campione di serie storiche simulate generate di lunghezza *N*, dove *N = 100, 250, 500, 750, 1000*, è stato applicato il metodo di differenziazione standard *'diffseries'* e il nuovo algoritmo proposto *'fracdiff'*, che sono due routine di R che si trovano nelle librerie *fracdiff*. Fatto questo, ad ognuna delle due serie storiche differenziate sono stati applicati i due diversi procedimenti di stima, precisamente prima lo stimatore *GPH* e poi lo stimatore di *Whittle*. Abbiamo poi calcolato alcune statistiche di nostro interesse, quali la distorsione (*bias*) e l'errore quadratico medio (*MSE*) per permetterci di stabilire se i due diversi algoritmi abbiano effettivamente differenziato le nostre serie in maniera adeguata.

I risultati ottenuti sono stati collocati all'interno di 4 diverse tabelle, ognuna per ogni diverso valore assegnato al parametro *d*. Ogni tabella è composta da 4 colonne, per ognuna delle quali sono riportati i valori delle nostre due statistiche calcolate. Nella prima colonna troviamo i risultati delle nostre statistiche dopo aver applicato ai campioni di serie storiche di partenza l'algoritmo di differenziazione standard '*diffseries*' e dopo

---

[3] Tale metodo di stima rientra sempre nel pacchetto del software R, chiamato *fracdiff*, da cui in precedenza abbiamo richiamato le funzioni per generare i campioni di serie storiche simulate e quelle dell'algoritmo *diffseries*.

aver applicato il metodo di stima *GPH*. Nella seconda colonna troviamo i valori delle statistiche dopo aver applicato ai campioni di serie storiche di partenza il nuovo algoritmo di differenziazione '*fracdiff*' e dopo aver applicato il metodo di stima *GPH*. Nella terza colonna invece troviamo i valori delle statistiche dopo aver applicato ai campioni di serie storiche di partenza l'algoritmo di differenziazione standard '*diffseries*' e dopo aver applicato lo stimatore di *Whittle*. Infine, nell'ultima colonna sono stati riportati i valori delle statistiche dopo aver applicato alle serie storiche di partenza il nuovo algoritmo di differenziazione '*fracdiff*' e dopo aver applicato lo stimatore di *Whittle*.

I dati contenuti nelle tabelle sono stati in seguito riportati nelle tabelle sottostanti:

Tabella 1

d=0.1	fdGPH(diffseries)		fdGPH(fracdiff)		whittle(diffseries)		whittle(fracdiff)	
	Bias	MSE	Bias	MSE	Bias	MSE	Bias	MSE
100	0.0106	0.0785	0.0106	0.0789	0.0213	0.0021	0.0213	0.0021
250	0.0103	0.0462	0.0102	0.0455	0.0142	0.0009	0.0142	0.0009
500	0.0052	0.0287	0.0056	0.0287	0.0107	0.0005	0.0107	0.0005
750	0.0031	0.0224	0.0035	0.0223	0.0082	0.0003	0.0082	0.0003
1000	0.0026	0.0171	0.0026	0.0172	0.0077	0.0002	0.0077	0.0002

Tabella 2

d=0.3	fdGPH(diffseries)		fdGPH(fracdiff)		whittle(diffseries)		whittle(fracdiff)	
	Bias	MSE	Bias	MSE	Bias	MSE	Bias	MSE
100	0.0157	0.0852	0.0194	0.0877	0.0197	0.0018	0.0204	0.0020
250	0.0080	0.0497	0.0105	0.0514	0.0137	0.0008	0.0143	0.0008
500	0.0054	0.0306	0.0088	0.0304	0.0114	0.0005	0.0118	0.0005
750	0.0049	0.0237	0.0084	0.0230	0.0098	0.0003	0.0099	0.0003
1000	0.0037	0.0189	0.0068	0.0188	0.0090	0.0002	0.0092	0.0002

Tabella 3

d=0.45	fdGPH(diffseries)		fdGPH(fracdiff)		whittle(diffseries)		whittle(fracdiff)	
	Bias	MSE	Bias	MSE	Bias	MSE	Bias	MSE
100	0.0121	0.0781	0.0495	0.0842	0.0205	0.0019	0.0293	0.0031
250	0.0120	0.0502	0.0468	0.0544	0.0155	0.0009	0.0210	0.0014
500	0.0103	0.0280	0.0424	0.0295	0.0111	0.0004	0.0159	0.0007
750	0.0075	0.0236	0.0351	0.0246	0.0098	0.0003	0.0131	0.0005
1000	0.0072	0.0192	0.0311	0.0203	0.0087	0.0002	0.0114	0.0004

Tabella 4

d=0.49	fdGPH(diffseries)		fdGPH(fracdiff)		whittle(diffseries)		whittle(fracdiff)	
	Bias	MSE	Bias	MSE	Bias	MSE	Bias	MSE
100	0.0252	0.0897	0.1525	0.1078	0.0213	0.0021	0.0651	0.0106
250	0.0148	0.0488	0.1349	0.0725	0.0153	0.0009	0.0510	0.0060
500	0.0083	0.0290	0.1199	0.0505	0.0122	0.0005	0.0415	0.0039
750	0.0079	0.0211	0.1155	0.0398	0.0106	0.0004	0.0334	0.0024
1000	0.0077	0.0194	0.1046	0.0382	0.0086	0.0003	0.0295	0.0019

3.4 Considerazioni

Come affermato in precedenza, le statistiche di cui ci serviamo per stabilire se i due algoritmi differenziano in maniera corretta sono la distorsione campionaria (*bias*) e l'errore quadratico medio (*MSE*). Per entrambe ci aspettiamo che i loro valori decrescano all'aumentare della numerosità campionaria. Inoltre, ci aspettiamo che la distorsione relativa al parametro *d*, sia prossima allo zero, poiché abbiamo appena svolto un procedimento di differenziazione delle nostre serie. Analizziamo e commentiamo ora i risultati ottenuti.

Consideriamo la ***Tabella 1***, contenente i valori ottenuti in seguito al procedimento svolto, dove l'iniziale valore del parametro di differenziazione è $d = 0.1$. Notiamo come

i valori della distorsione siano prossimi allo zero, e decrescano ulteriormente all'aumentare della numerosità campionaria. Questa osservazione è valida in tutte le 4 colonne della tabella. Osserviamo l'errore quadratico medio: in questo caso notiamo come quest'ultimo sia molto più alto nelle serie stimate con lo stimatore *GPH*, il che ci fa preferire senza alcun dubbio, il metodo di stima di *Whittle*, il quale presenta errori quadratici medi molto piccoli, indifferentemente dall'algoritmo di differenziazione applicato in precedenza. In conclusione, per un valore di $d = 0.1$, possiamo affermare che i due algoritmi differenziano in maniera adeguata; dunque, se potessimo scegliere, sceglieremmo senz'altro il nuovo algoritmo proposto (*'fracdiff'*) in quanto riduce sostanzialmente il costo computazionale.

Consideriamo ora la ***Tabella 2***, contenente i risultati dell'analisi fatta assegnando al parametro di differenziazione *d* il valore di *0.3*. Anche in questo caso possiamo notare come i valori della distorsione siano molto vicini allo zero, e decrescano ulteriormente con l'aumentare della numerosità campionaria. Notiamo inoltre, come nel caso precedente, che il metodo di stima attraverso lo stimatore di *Whittle* è preferibile allo stimatore *GPH*, in quanto l'errore quadratico medio nelle serie stimate con il primo metodo è decisamente minore rispetto a quelle stimate con il secondo metodo. Comunque, per quanto riguarda l'adeguata differenziazione dei due algoritmi *'diffseries'* e *'fracdiff'*, anche in questo caso possiamo affermare che essi hanno differenziato correttamente i diversi campioni di serie storiche. Dunque, a parità di condizioni, saremmo indirizzati a scegliere l'algoritmo *'fracdiff'* basato sulla trasformata veloce di Fourier, poiché riduce notevolmente il costo computazionale, come abbiamo visto nel nostro studio principale.

Passiamo ora alla ***Tabella 3***, dove troviamo i risultati ottenuti dall'analisi sui campioni di serie storiche differenziate, dove il parametro di differenziazione è $d = 0.45$. Come fatto precedentemente, controlliamo i valori della distorsione nelle diverse colonne. Sebbene siano leggermente più elevati, i valori ottenuti si possono considerare buoni, poiché sono abbastanza prossimi allo zero e decrescono ulteriormente all'aumentare della numerosità campionaria. Anche in questo caso, come nei due precedenti, notiamo come lo stimatore *GPH* non stimi molto bene le serie differenziate, infatti presenta sempre *MSE* piuttosto

elevati, contrariamente invece allo stimatore di *Whittle*, i cui *MSE* vanno molto bene. Per quanto riguarda dunque i due algoritmi di differenziazione, possiamo anche in questo caso considerarli equivalenti al fine di differenziare dei campioni di serie storiche, in quanto le differenziano in maniera adeguata. Privilegeremo comunque il nuovo algoritmo proposto, molto più veloce(dell'ordine di *TlogT*) rispetto all'algoritmo standard(dell'ordine di *T^2*).

Infine consideriamo l'ultima delle quattro tabelle (***Tabella 4***), quella inerente l'analisi sui campioni di serie storiche differenziate, il cui parametro di differenziazione è *d = 0.49*. Notiamo di trovarci quasi al limite della soglia di stazionarietà, soglia oltre la quale il nostro processo non è più stazionario. Osserviamo ed analizziamo i valori della distorsione come nei precedenti casi. Questa volta però, la nostra attenzione cade principalmente sulla seconda e quarta colonna, dove i campioni di serie storiche sono stati differenziati utilizzando il nuovo algoritmo proposto. I valori della distorsione sono davvero elevati, basti osservare i valori nella seconda colonna, dove per campioni di lunghezza *100* la distorsione relativa al abbiamo un parametro di differenziazione è pari a *0.1525*, e non circa zero come ci aspetteremmo. Per quanto riguarda invece i campioni differenziati con l'algoritmo standard, i valori sono meno elevati e decrescono verso lo zero all'aumentare della numerosità campionaria. Per quanto riguarda gli *MSE*, notiamo valori anomali specialmente per la seconda e quarta colonna, ovvero le colonne differenziate col nuovo metodo proposto. Inoltre notiamo, come al solito, come lo stimatore di *Whittle* sia migliore dello stimatore *GPH*. In conclusione, in questo caso non possiamo dire che i due algoritmi differenziano le serie entrambi in maniera adeguata, poiché il nostro algoritmo proposto sembra non comportarsi correttamente. Probabilmente, ciò è dettato dal fatto che abbiamo scelto un valore del parametro *d* molto vicino alla soglia di non stazionarietà, il che rende meno efficace il nostro algoritmo. In questa situazione dunque, conviene scegliere l'algoritmo standard (*'diffseries'*) per differenziare i nostri campioni, in quanto ci dà la certezza di differenziare in maniera adeguata.

CONCLUSIONI

In questo elaborato, abbiamo fornito un algoritmo veloce per il calcolo della differenza frazionaria di una serie storica basata sul teorema di convoluzione circolare e la trasformata veloce di Fourier. Il numero richiesto di operazioni aritmetiche per il nuovo algoritmo è dell'ordine di $TlogT$, comparato con T^2 per le implementazioni standard, e similarmente per il tempo computazionale.

Per campioni di grandezza medio-grandi, la differenza di tempo in termini computazionali è molto sostanziale e può facilmente essere la differenza tra una stima realizzabile e una non realizzabile. Per di più, il calcolo più veloce della differenza frazionaria realizzato dal nuovo algoritmo apre nuove possibilità per l'applicazione di metodi bootstrap o metodi di simulazione a serie storiche generate da modelli *ARFIMA* di medio-grandi dimensioni.

Notiamo che la parte computazionale intensa nell'algoritmo è il calcolo della trasformata discreta di Fourier. Quando la differenza frazionaria è applicata a serie storiche multiple, i coefficienti della serie hanno solo bisogno di essere trasformati una volta, e di conseguenza, il nostro algoritmo scala bene con il numero di variabili. In più, la trasformata veloce di Fourier può essere calcolata in parallelo. Nelle nostre simulazioni, abbiamo esplicitamente non preso vantaggio di questa caratteristica, così da ottenere un confronto più ragionevole con l'implementazione standard. Comunque, in pratica calcolare la trasformata veloce di Fourier in parallelo renderà il nostro algoritmo perfino più veloce sulla maggior parte dei computer moderni.

I risultati prodotti attraverso il nostro studio di simulazione sono in linea con quelli ottenuti nell'articolo "A fast fractional difference algorithm" di Andreas Noack Jensen e Morten Ørregaard Nielsen (2014).

Inoltre, per quanto riguarda l'ulteriore studio fatto sui due diversi algoritmi, concludiamo affermando come entrambi gli algoritmi differenzino in maniera adeguata serie storiche che hanno valori del parametro di differenziazione d non troppo vicini alla soglia di non stazionarietà. Per quegli altri valori infatti, è preferibile utilizzare l'algoritmo standard,

seppur molto dispendioso dal punto di vista computazionale, in quanto l'algoritmo basato sulla trasformata veloce di Fourier non differenzia in maniera adeguata come in precedenza.

Appendice: Codice sorgente R

```
library(fracdiff)
# PGD: ARFIMA
N = seq(100,100000, by=100)
l=length(N)
d=0.1
M05<- matrix(0,ncol=5,nrow=l)

# Introduciamo la routine fracdiff, che rappresenta l'algoritmo basato sulla trasformata veloce di Fourier
fracdiff<-function(x,d){
iT<-length(x)
np2<-nextn(2*iT-1,2)
k<-1:(iT-1)
b<-c(1,cumprod((k-d-1)/k))
dx<-fft(fft(c(b,rep(0,np2-iT)))*fft(c(x,rep(0,np2-iT))), inverse=T)/np2;
return(Re(dx[1:iT]))
}

# Introduciamo la library(microbenchmark), da cui utilizziamo una funzione per calcolare i tempi impiegati dai due algoritmi per differenziare i nostri campioni
library(microbenchmark)

# Calcoliamo i tempi impiegati dai due algoritmi per campioni di serie con una numerosità che va da 100 fino a 100.000, e collochiamo i risultati all'interno di una matrice
for (i in 1:l) {
x=fracdiff.sim(N[i], d=d)
t1 <- microbenchmark(diffseries(x$series,x$d),times=1)$time/10^9
t2 <- microbenchmark(fracdiff(x$series,x$d),times=1)$time/10^9
M01[i,] <- cbind(N[i],t1,t2)
}
```

 # Rappresentiamo graficamente i risultati ottenuti

matplot(log(M01[,1]),log(M01[,2:3]),type="l",xlab="samplesize",ylab="Seconds",main="Grafico")

legend(“topleft”, legend=c(“diffseries”,”fracdiff”), col=c(1,2), lty=1)

 # Stima del parametro d con i metodi "fdGPH" e "whittle"

 # Introduciamo la routine per poter utilizzare lo stimatore di Whittle

#Stima del periodogramma

```
periodogram<-function(x)
{
t <- length(x)
tt <- trunc((t-1)/2)
freq <- array(1,tt)
y <- array(1,tt)
i <- c(1:t)
for (j in 1:tt)
{
cos_j <- cos(2*pi*j*i/t)
sen_j <- sin(2*pi*j*i/t)
y[j] <- ((cos_j%*%x)^2 + (sen_j%*%x)^2)/(pi*t)
freq[j] <- 2*pi*j/t
}
return (y)
}
```

#Calcolo dello spettro

```
spettro<-function(t)
{
tt<-trunc((t-1)/2)
y<-array (1,tt)
i<-c(1:t)
for (j in 1:tt)
{
y [j]<- abs(2*sin(pi*j/t))
}
return(y)
```

```
}
whittle<-function(x)
{
result <- list()
#d<-real
#dhat<-real
t <- length(x)
period <- periodogram(x)
spec <- spettro(t)
q<-function(d) t(period)%*%spec^(2*d)
min<-optimize(q,c(0,1))
dhat<-min$minimum
result$d <- dhat
return(result)
}
```

 # Stimiamo ora il parametro d con fdGPH e Whittle e poniamo i risultati all'interno di una matrice

```
s=1000
M_100<- matrix(0,ncol=4,nrow=s)
d=0.1
for (j in 1:s){
x=fracdiff.sim(100, d=d)
d1a <-fdGPH((diffseries(x$series,x$d)), bandw.exp = 0.5)[1] $d
d1b <-fdGPH((fracdiff(x$series,x$d)), bandw.exp = 0.5)[1] $d
d1c <-whittle(diffseries(x$series,x$d))$d
d1d <-whittle(fracdiff(x$series,x$d))$d
M_100[j,] <- cbind(d1a,d1b,d1c,d1d)
}
```

 # Calcoliamo la distorsione per ogni colonna della matrice

```
bias = apply(M_100,2,mean)
bias
```

 # Calcoliamo l'errore quadratico medio per ogni colonna della matrice

```
di <- M_100[,1]
MSE <- mean((di)^2)
MSE
di <- M_100[,2]
MSE <- mean((di)^2)
MSE
di <- M_100[,3]
MSE <- mean((di)^2)
MSE
di <- M_100[,4]
MSE <- mean((di)^2)
MSE
```

 # Ripeteremo dall'inizio le stesse routine assegnando al parametro di differenziazione d i valori 0.3, 0.45, 0.49

Bibliografia

Azzalini, A. 2001. *Inferenza statistica: una presentazione basata sul concetto di verosimiglianza.* Milano. Springer.

Beran J. 1994. *Statistics for Long-Memory Processes*. London. England. Chapman and Hall.

Cooley JW, Lewis PAW, Welch PD. 1969. *The fast Fourier transform and its applications.* IEEE Transactions on Education.

Di Fonzo, T. and Lisi, F. 2005. *Serie storiche economiche – Analisi statistiche e applicazioni.* Carrocci editore.

Jensen, Andreas N. and Nielsen, Morten Ø. 2014. A fast fractional difference algorithm. *Journal of time series analysis,* 35: 428-436. 2014.

Maechler M. 2012. *The fracdiff package for R, version 1.4-2*.

Palma W. 2007. *Long-Memory Time Series: Theory and Methods.* New Jersey: John Wiley and Sons.

Shumway, Robert H. and Stoffer, David S. 2011. *Time Series Analysis and Its Application*. Springer.

Whittle, P. 1951. *Hypothesis Testing in Time Series Analysis*. Hafner.