Metodi basati su matrici strutturate per il calcolo del GCD approssimato di polinomi

P. Boito

(Scuola Normale Superiore, Pisa)

Lavoro in collaborazione con D. Bini

Giornate di Algebra Lineare Numerica, 26-27 febbraio 2007

Sommario

Motivazione

Problema

Dati $u(x), v(x) \in \mathbb{R}[x]$ *o* $\mathbb{C}[x]$, *calcolare* $g(x) = GCD(u, v)$.

- Applicazioni: calcolo di radici di polinomi, teoria del controllo, CAGD, image deblurring...
- Ma se i coefficienti di $u(x)$ and $v(x)$ sono affetti da errore, il problema è mal definito.
- Occorre una nuova definizione di GCD polinomiale [Sch85].

Motivazione

Problema

Dati $u(x), v(x) \in \mathbb{R}[x]$ *o* $\mathbb{C}[x]$, *calcolare* $g(x) = GCD(u, v)$.

- Applicazioni: calcolo di radici di polinomi, teoria del controllo, CAGD, image deblurring...
- Ma se i coefficienti di $u(x)$ and $v(x)$ sono affetti da errore, il problema è mal definito.
- Occorre una nuova definizione di GCD polinomiale [Sch85].

Motivazione

Problema

Dati $u(x), v(x) \in \mathbb{R}[x]$ *o* $\mathbb{C}[x]$, *calcolare* $g(x) = GCD(u, v)$.

- Applicazioni: calcolo di radici di polinomi, teoria del controllo, CAGD, image deblurring...
- Ma se i coefficienti di $u(x)$ and $v(x)$ sono affetti da errore, il problema è mal definito.
- Occorre una nuova definizione di GCD polinomiale [Sch85].

Motivazione

Problema

Dati $u(x), v(x) \in \mathbb{R}[x]$ *o* $\mathbb{C}[x]$, *calcolare* $g(x) = GCD(u, v)$.

- Applicazioni: calcolo di radici di polinomi, teoria del controllo, CAGD, image deblurring...
- Ma se i coefficienti di $u(x)$ and $v(x)$ sono affetti da errore, il problema è mal definito.
- Occorre una nuova definizione di GCD polinomiale [Sch85].

ϵ-GCD

Definizione

Dato $\epsilon > 0$,

- $g(x)$ *è un* ϵ*-divisore di* $u(x)$ *e* $v(x)$ *se è un divisore esatto di* $\hat{u}(x)$ *e* $\hat{v}(x)$*, dove:*
  - $\hat{u}(x)$ *e* $\hat{v}(x)$ *hanno gli stessi gradi di* $u(x)$ *e* $v(x)$*;*
  - $\|u(x) - \hat{u}(x)\|_2 < \epsilon, \qquad \|v(x) - \hat{v}(x)\|_2 < \epsilon.$
- $g(x)$ *è un* ϵ*-GCD di* $u(x)$ *e* $v(x)$ *se è un* ϵ*-divisore di grado massimo.*

Il grado dell' ϵ-GCD è unicamente definito, mentre i coefficienti non lo sono.

ϵ-GCD

Definizione

Dato $\epsilon > 0$,

- $g(x)$ *è un* ϵ*-divisore di* $u(x)$ *e* $v(x)$ *se è un divisore esatto di* $\hat{u}(x)$ *e* $\hat{v}(x)$*, dove:*
  - $\hat{u}(x)$ *e* $\hat{v}(x)$ *hanno gli stessi gradi di* $u(x)$ *e* $v(x)$*;*
  - $\|u(x) - \hat{u}(x)\|_2 < \epsilon, \qquad \|v(x) - \hat{v}(x)\|_2 < \epsilon.$
- $g(x)$ *è un* ϵ*-GCD di* $u(x)$ *e* $v(x)$ *se è un* ϵ*-divisore di grado massimo.*

Il grado dell' ϵ-GCD è unicamente definito, mentre i coefficienti non lo sono.

ϵ-GCD

Definizione

Dato $\epsilon > 0$,

- *$g(x)$ è un ϵ-divisore di $u(x)$ e $v(x)$ se è un divisore esatto di $\hat{u}(x)$ e $\hat{v}(x)$, dove:*
  - *$\hat{u}(x)$ e $\hat{v}(x)$ hanno gli stessi gradi di $u(x)$ e $v(x)$;*
  - *$\|u(x) - \hat{u}(x)\|_2 < \epsilon, \qquad \|v(x) - \hat{v}(x)\|_2 < \epsilon.$*
- *$g(x)$ è un ϵ-GCD di $u(x)$ e $v(x)$ se è un ϵ-divisore di grado massimo.*

Il grado dell' ϵ-GCD è unicamente definito, mentre i coefficienti non lo sono.

ϵ-GCD

Definizione

Dato $\epsilon > 0$,

- $g(x)$ *è un* ϵ*-divisore di* $u(x)$ *e* $v(x)$ *se è un divisore esatto di* $\hat{u}(x)$ *e* $\hat{v}(x)$*, dove:*
  - $\hat{u}(x)$ *e* $\hat{v}(x)$ *hanno gli stessi gradi di* $u(x)$ *e* $v(x)$*;*
  - $\|u(x) - \hat{u}(x)\|_2 < \epsilon, \qquad \|v(x) - \hat{v}(x)\|_2 < \epsilon.$
- $g(x)$ *è un* ϵ*-GCD di* $u(x)$ *e* $v(x)$ *se è un* ϵ*-divisore di grado massimo.*

Il grado dell' ϵ-GCD è unicamente definito, mentre i coefficienti non lo sono.

Matrice di Sylvester

- Siano $u(x) = \sum_{i=0}^{n} u_i x^i, \qquad v(x) = \sum_{i=0}^{m} v_i x^i.$
- Si definisce

$$\mathrm{Sylv}(u,v) = \begin{pmatrix} u_n & u_{n-1} & \cdots & \cdots & u_0 & & \\ & \ddots & \ddots & & \ddots & & \\ & & u_n & u_{n-1} & \cdots & \cdots & u_0 \\ v_m & v_{m-1} & \cdots & v_0 & & & \\ & \ddots & \ddots & & \ddots & & \\ & & \ddots & \ddots & & \ddots & \\ & & & v_m & v_{m-1} & \cdots & v_0 \end{pmatrix}.$$

- $\mathrm{Sylv}(u,v)$ è una matrice quadrata di ordine $n+m$.

Matrice di Bézout

- Siano $u(x) = \sum_{i=0}^{n} u_i x^i, \qquad v(x) = \sum_{i=0}^{m} v_i x^i, \qquad n \geq m.$ Allora

$$b(x,y) = \frac{u(x)v(y) - u(y)v(x)}{x-y} = \sum_{i,j} b_{ij} x^i y^j$$

  è un polinomio in due variabili.
- Si definisce $(\mathrm{Bez}(u,v))_{ij} = b_{ij}$.
- $\mathrm{Bez}(u,v)$ è una matrice quadrata e simmetrica, di ordine n.

Matrice di Bézout

- Siano $u(x) = \sum_{i=0}^{n} u_i x^i, \qquad v(x) = \sum_{i=0}^{m} v_i x^i, \qquad n \geq m.$ Allora

$$b(x,y) = \frac{u(x)v(y) - u(y)v(x)}{x-y} = \sum_{i,j} b_{ij} x^i y^j$$

  è un polinomio in due variabili.
- Si definisce $(\mathrm{Bez}(u,v))_{ij} = b_{ij}$.
- $\mathrm{Bez}(u,v)$ è una matrice quadrata e simmetrica, di ordine n.

Matrice di Bézout

- Siano $u(x) = \sum_{i=0}^{n} u_i x^i, \qquad v(x) = \sum_{i=0}^{m} v_i x^i, \qquad n \geq m.$ Allora

$$b(x,y) = \frac{u(x)v(y) - u(y)v(x)}{x-y} = \sum_{i,j} b_{ij} x^i y^j$$

  è un polinomio in due variabili.
- Si definisce $(\mathrm{Bez}(u,v))_{ij} = b_{ij}$.
- $\mathrm{Bez}(u,v)$ è una matrice quadrata e simmetrica, di ordine n.

Proprietà delle matrici risultanti

Sia $M = \mathrm{Sylv}(u, v)$ oppure $\mathrm{Bez}(u, v)$.

- $\dim \mathrm{Ker}\,(M) = \deg \mathrm{GCD}\,(u, v)$;
- sia $M = QR$, dove

$$R = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 \end{pmatrix};$$

allora l'ultima riga non nulla di R fornisce un GCD di $u(x)$ e $v(x)$;

- M è Toeplitz-like con displacement rank 2.

Proprietà delle matrici risultanti

Sia $M = \mathrm{Sylv}(u, v)$ oppure $\mathrm{Bez}(u, v)$.

- dim Ker $(M) =$ deg GCD (u, v);
- sia $M = QR$, dove

$$R = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 \end{pmatrix};$$

  allora l'ultima riga non nulla di R fornisce un GCD di $u(x)$ e $v(x)$;
- M è Toeplitz-like con displacement rank 2.

Proprietà delle matrici risultanti

Sia $M = \mathrm{Sylv}(u, v)$ oppure $\mathrm{Bez}(u, v)$.

- dim Ker $(M) =$ deg GCD (u, v);
- sia $M = QR$, dove

$$R = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 \end{pmatrix};$$

allora l'ultima riga non nulla di R fornisce un GCD di $u(x)$ e $v(x)$;

- M è Toeplitz-like con displacement rank 2.

Proprietà delle matrici risultanti

Sia $M = \mathrm{Sylv}(u, v)$ oppure $\mathrm{Bez}(u, v)$.

- dim Ker $(M) =$ deg GCD (u, v);
- sia $M = QR$, dove

$$R = \begin{pmatrix} \times & \times & \times & \times \\ 0 & \times & \times & \times \\ 0 & 0 & \times & \times \\ 0 & 0 & 0 & 0 \end{pmatrix};$$

  allora l'ultima riga non nulla di R fornisce un GCD di $u(x)$ e $v(x)$;
- M è Toeplitz-like con displacement rank 2.

Matrice di Sylvester e coefficienti del GCD

$$g(x) = \mathrm{GCD}(u, v) \Leftrightarrow \begin{cases} u(x) = p(x)g(x) \\ v(x) = q(x)g(x) \end{cases} \Leftrightarrow u(x)q(x) = v(x)p(x).$$

In forma matriciale:

$$S = \begin{bmatrix} \mathbf{q} \\ -\mathbf{p} \end{bmatrix} = 0, \quad S = \begin{pmatrix} u_0 & & 0 & v_0 & & 0 \\ \vdots & \ddots & & \vdots & \ddots & \\ u_n & & u_0 & v_m & & v_0 \\ & \ddots & \vdots & & \ddots & \vdots \\ 0 & & u_n & 0 & & v_m \end{pmatrix}.$$

Matrice di Sylvester e coefficienti del GCD

$$g(x) = \mathrm{GCD}(u, v) \Leftrightarrow \begin{cases} u(x) = p(x)g(x) \\ v(x) = q(x)g(x) \end{cases} \Leftrightarrow u(x)q(x) = v(x)p(x).$$

In forma matriciale:

$$S = \begin{bmatrix} \mathbf{q} \\ -\mathbf{p} \end{bmatrix} = 0, \quad S = \begin{pmatrix} u_0 & & 0 & v_0 & & 0 \\ \vdots & \ddots & & \vdots & \ddots & \\ u_n & & u_0 & v_m & & v_0 \\ & \ddots & \vdots & & \ddots & \vdots \\ 0 & & u_n & 0 & & v_m \end{pmatrix}.$$

Idea per un algoritmo

Come calcolare un ϵ-GCD:

1. stima su deg ϵ-GCD ottenuta valutando il rango approssimato di Sylv(u, v) o Bez(u, v);
2. calcolo dei coefficienti (sistema definito da una sottomatrice di Sylv(u, v) o Bez(u, v));
3. raffinamento iterativo (Newton).

Obiettivo: fare questo in modo stabile e con costo quadratico.

Idea per un algoritmo

Come calcolare un ϵ-GCD:

1. stima su deg ϵ-GCD ottenuta valutando il rango approssimato di Sylv(u, v) o Bez(u, v);
2. calcolo dei coefficienti (sistema definito da una sottomatrice di Sylv(u, v) o Bez(u, v));
3. raffinamento iterativo (Newton).

Obiettivo: fare questo in modo stabile e con costo quadratico.

Idea per un algoritmo

Come calcolare un ϵ-GCD:

1. stima su deg ϵ-GCD ottenuta valutando il rango approssimato di Sylv(u, v) o Bez(u, v);
2. calcolo dei coefficienti (sistema definito da una sottomatrice di Sylv(u, v) o Bez(u, v));
3. raffinamento iterativo (Newton).

Obiettivo: fare questo in modo stabile e con costo quadratico.

Idea per un algoritmo

Come calcolare un ϵ-GCD:

1. stima su deg ϵ-GCD ottenuta valutando il rango approssimato di Sylv(u, v) o Bez(u, v);
2. calcolo dei coefficienti (sistema definito da una sottomatrice di Sylv(u, v) o Bez(u, v));
3. raffinamento iterativo (Newton).

Obiettivo: fare questo in modo stabile e con costo quadratico.

Matrici Toeplitz-like

$T \in \mathbb{C}^{n \times n}$ è Toeplitz-like con displacement rank r se

$$\nabla_{\{1,-1\}}(T) = Z_1 \cdot T - T \cdot Z_{-1} = G \cdot B,$$

dove $G \in \mathbb{C}^{n \times r}$, $B \in \mathbb{C}^{r \times n}$ sono i generatori, e

$$Z_\phi = \begin{pmatrix} 0 & \dots & \dots & 0 & \phi \\ 1 & 0 & \dots & \dots & 0 \\ 0 & 1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \vdots \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix}.$$

Matrici Cauchy-like

- $C \in \mathbb{C}^{n\times n}$ è Cauchy-like con displacement rank r se

$$\nabla_{\{F,A\}}(C) = F \cdot C + C \cdot A^* = G \cdot B,$$

dove $G \in \mathbb{C}^{n\times r}$, $B \in \mathbb{C}^{r\times n}$ sono i generatori e $F = \mathrm{diag}(f_0, f_1, \ldots, f_{n-1})$, $A = \mathrm{diag}(a_0, a_1, \ldots, a_{n-1})$ sono matrici diagonali con spettri distinti.

- Si ha

$$C = \left[\frac{\mathbf{g}_i \mathbf{b}_j^*}{f_i - \bar{a}_j}\right]_{i,j=0}^{n-1}.$$

Matrici Cauchy-like

- $C \in \mathbb{C}^{n\times n}$ è Cauchy-like con displacement rank r se

$$\nabla_{\{F,A\}}(C) = F \cdot C + C \cdot A^* = G \cdot B,$$

dove $G \in \mathbb{C}^{n\times r}$, $B \in \mathbb{C}^{r\times n}$ sono i generatori e $F = \mathrm{diag}(f_0, f_1, \ldots, f_{n-1})$, $A = \mathrm{diag}(a_0, a_1, \ldots, a_{n-1})$ sono matrici diagonali con spettri distinti.

- Si ha

$$C = \left[\frac{\mathbf{g}_i \mathbf{b}_j^*}{f_i - \bar{a}_j}\right]_{i,j=0}^{n-1}.$$

Fast LU con pivoting parziale (GKO)

Sia $C \in \mathbb{C}^{n \times n}$ Cauchy-like. È possibile applicare a C l'eliminazione gaussiana con pivoting parziale con un costo computazionale di $\mathcal{O}(n^2)$ ([GKO95]). Infatti:

- un passo di eliminazione gaussiana equivale a calcolare un complemento di Schur,
- il complemento di Schur conserva la struttura della matrice di partenza,
- permutando le righe di C, si mantiene la struttura Cauchy-like,
- di conseguenza è possibile lavorare sui generatori anziché sull'intera matrice.

Fast LU con pivoting parziale (GKO)

Sia $C \in \mathbb{C}^{n \times n}$ Cauchy-like. È possibile applicare a C l'eliminazione gaussiana con pivoting parziale con un costo computazionale di $\mathcal{O}(n^2)$ ([GKO95]). Infatti:

- un passo di eliminazione gaussiana equivale a calcolare un complemento di Schur,
- il complemento di Schur conserva la struttura della matrice di partenza,
- permutando le righe di C, si mantiene la struttura Cauchy-like,
- di conseguenza è possibile lavorare sui generatori anziché sull'intera matrice.

Toeplitz-like $\longrightarrow$ Cauchy-like

Teorema

Sia $T \in \mathbb{C}^{n\times n}$ Toeplitz-like con generatori G, B. Allora ([Heinig94])

$$C = \mathcal{F} T D_0^{-1} \mathcal{F}^*$$

è Cauchy-like, ovvero

$$\nabla_{D_1, D_{-1}}(C) = D_1 C - C D_{-1} = \hat{G}\hat{B},$$

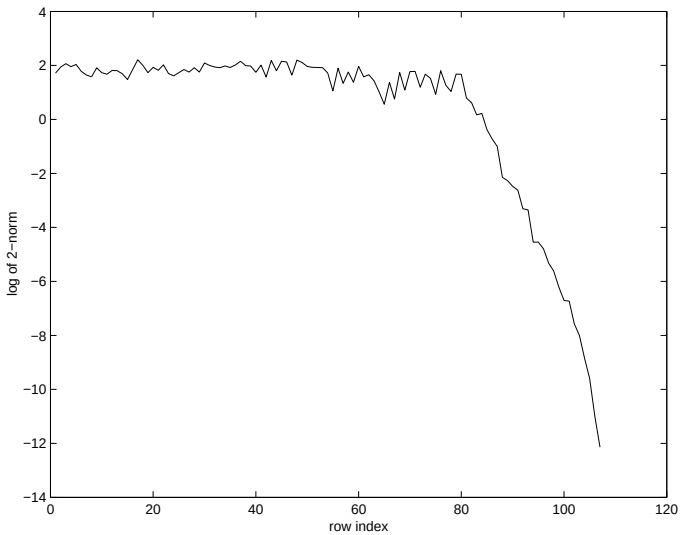
dove $\mathcal{F}$ è la matrice di Fourier, D_0, D_1, D_{-1} sono opportune matrici diagonali e

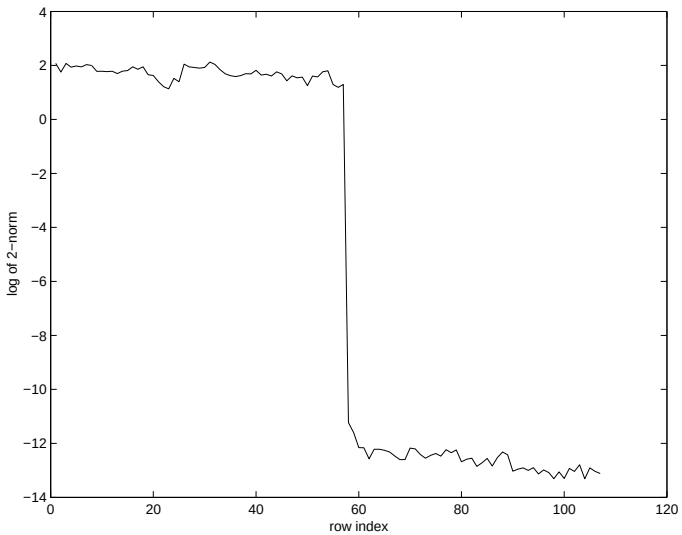
$$\hat{G} = \mathcal{F} G, \qquad \hat{B}^* = \mathcal{F} D_0 B^*.$$

Modifiche a GKO: pivoting

- GKO è instabile sulle matrici risultanti (crescita dei generatori).
- Calcolando $\mathrm{Sylv}(u, v) = \Pi LU$ con GKO, le norme delle righe di U decrescono gradualmente, anziché mostrare un "salto" netto.
- Il problema si può risolvere ortogonalizzando ad ogni passo il primo generatore e applicando un pivoting opportuno sul secondo.

GKO e rango approssimato





Modifiche a GKO: matrici rettangolari

Sia $T \in \mathbb{C}^{n \times m}$, $n \neq m$, una matrice Toeplitz-like con displacement rank r.
La riduzione di T a Cauchy-like va riformulata usando il fatto che

$$\nabla_\theta(T) = Z_1 \cdot T - T \cdot Z_\theta$$

ha rango r per ogni $\theta \in \mathbb{C}$ tale che $|\theta| = 1$.
È quindi possibile calcolare i coefficienti del GCD (noto il grado) in modo stabile e con costo quadratico.

Determinazione del grado

Una stima sul grado dell'ϵ-GCD si può ottenere:

- per bisezione, oppure
- usando un criterio euristico: un limite superiore sul grado è dato da $n - k_\epsilon$, dove

$$k_\epsilon = \max\{k : a_k < \epsilon\sqrt{n+m}\}$$

e a_k è il k-esimo pivot calcolato nella fattorizzazione della matrice Cauchy-like ottenuta da Sylv(u, v).

Raffinamento iterativo

Il sistema

$$\begin{cases} u(x) = g(x)p(x) \\ v(x) = g(x)q(x) \end{cases}$$

può essere scritto come

$$A(\mathbf{z}) - \mathbf{w} = 0,$$

dove

$$\mathbf{z} = \begin{bmatrix} \mathbf{g} \\ \mathbf{p} \\ \mathbf{q} \end{bmatrix}, \mathbf{w} = \begin{bmatrix} \mathbf{u} \\ \mathbf{v} \end{bmatrix}, A(\mathbf{z}) = \begin{bmatrix} C(p)\mathbf{g} \\ C(q)\mathbf{g} \end{bmatrix} = \begin{bmatrix} C(g)\mathbf{p} \\ C(g)\mathbf{q} \end{bmatrix}$$

e $C(p)$, $C(q)$, $C(g)$ sono matrici di convoluzione.

Raffinamento iterativo

Metodo di Newton: posto $F(\mathbf{z}) = A(\mathbf{z}) - \mathbf{w}$, ad ogni passo deve essere risolto il problema lineare ai minimi quadrati

$$J(\mathbf{z_j} - \mathbf{z_{j+1}}) = F(\mathbf{z_j}),$$

dove J è lo jacobiano associato a $F(\mathbf{z})$ ed è una matrice Toeplitz-like.
Il metodo GKO modificato può essere utilizzato per calcolare una successione $\{\mathbf{z}_j\}$ che converge con la stessa velocità del metodo di Newton "classico".

L'algoritmo Fastgcd

Input: polinomi $u(x)$ e $v(x)$, una tolleranza ϵ.
Output: un ϵ-GCD, cofattori, residuo (backward error).

1. Determina una stima k di deg ϵ-GCD.
2. Calcola i coefficienti di un divisore approssimato $g(x)$ di grado k, con relativi cofattori.
3. Esegue il raffinamento iterativo.
4. $g(x)$ è un ϵ-divisore?
5. Se sì, pone k=k+1 e calcola un nuovo divisore approssimato, fino a quando viene trovato un ϵ-divisore di grado massimo.
6. Se no, pone k=k-1 e calcola un nuovo divisore approssimato, fino a quando viene trovato un ϵ-divisore.

Grado variabile (1)

- ([Zeng]) Siano

$$u(x) = \prod_{1}^{10}(x - x_j),$$

$$v(x) = \prod_{1}^{10}(x - x_j + 10^{-j}),$$

con $x_j = (-1)^j(j/2)$.

- Le radici di $u(x)$ e $v(x)$ hanno distanze decrescenti pari a 0.1, 0.01, ecc.

Grado variabile (2)

ϵ	Fastgcd		UVGCD	
	deg	res	deg	res
10^{-2}	9	0.0045	9	0.0041
10^{-3}	8	2.63×10^{-4}	8	1.73×10^{-4}
10^{-4}	7	9.73×10^{-6}	8	1.73×10^{-4} $(*)$
10^{-5}			4	1.80×10^{-5}
10^{-6}	5	8.59×10^{-9}	2	2.25×10^{-14}
10^{-7}				
10^{-8}				
10^{-9}	1	3.98×10^{-11}		

Radici multiple

Per $k \in \mathbb{N}$ siano $u(x) = (x^3 + 3x - 1)(x - 1)^k$ e $v(x) = u'(x)$.
La tabella mostra i residui (backward error) e gli errori coefficient-wise sul GCD calcolato rispetto a quello teorico.

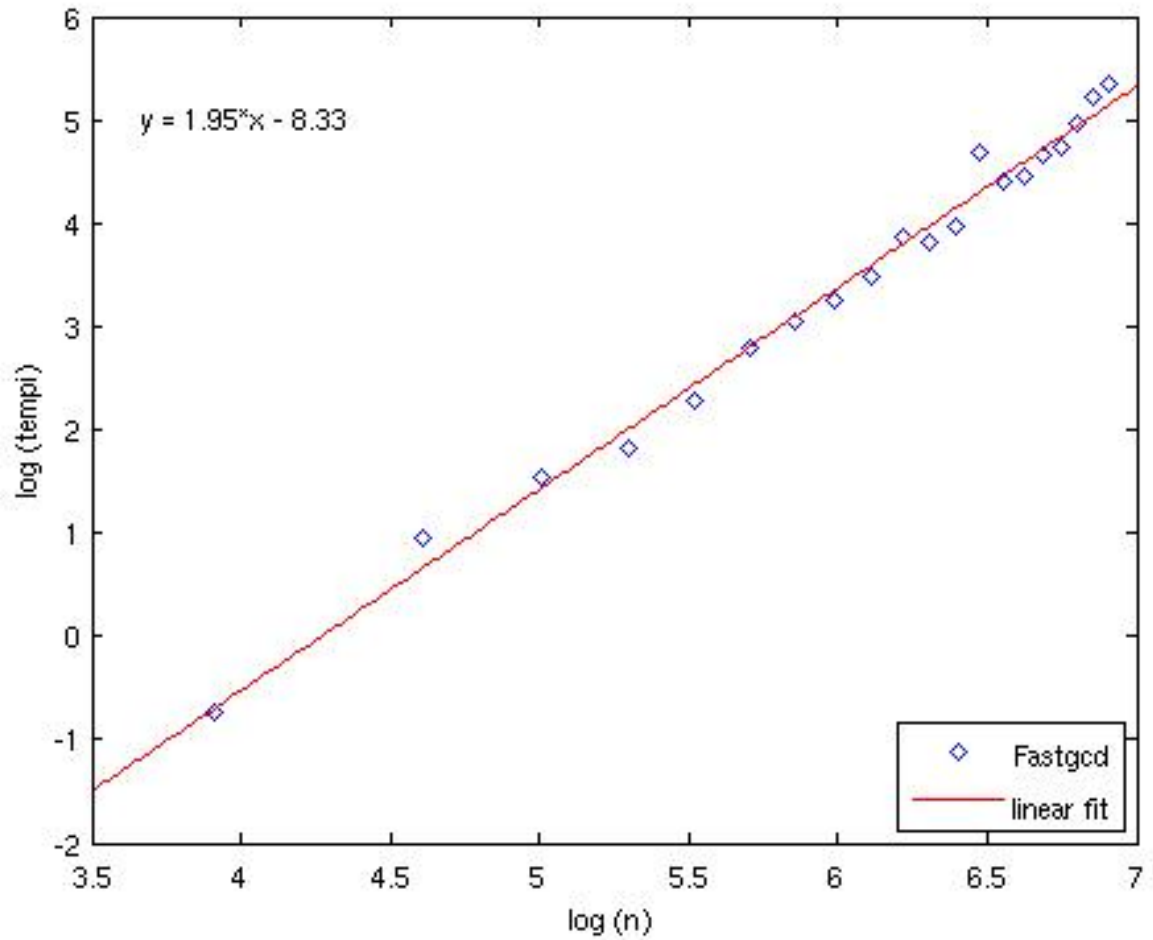
k	res	err
15	1.72×10^{-16}	1.40×10^{-13}
25	2.14×10^{-16}	1.14×10^{-10}
35	1.21×10^{-16}	1.36×10^{-8}
45	1.31×10^{-16}	1.85×10^{-5}
55	7.49×10^{-16}	7.50×10^{-3}

Costo quadratico (1)

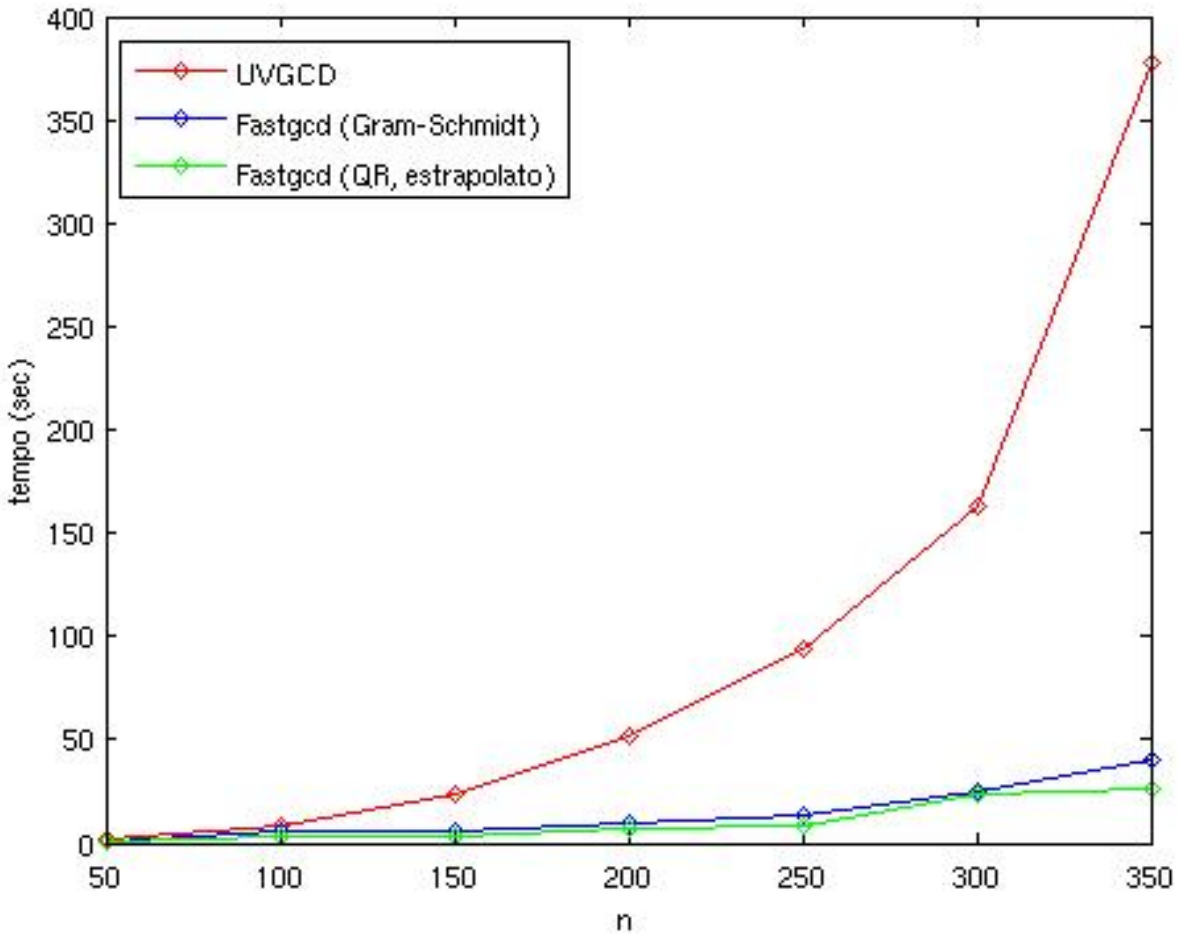
Per $k \in \mathbb{N}$, siano $n_1 = 25k$, $n_2 = 15k$, $n_3 = 10k$. Si definiscono $u_k(x) = g(x)p_k(x)$ e $v_k(x) = g(x)q_k(x)$, dove

$$p_k(x) = (x^{n_1} - 1)(x^{n_2} - 2)(x^{n_3} - 3),$$
$$q_k(x) = (x^{n_1} + 1)(x^{n_2} + 5)(x^{n_3} + i),$$
$$g(x) = x^4 + 10x^3 + x - 1.$$

Costo quadratico (2)



Confronto con UVGCD



Conclusioni

- È stato progettato e implementato un nuovo algoritmo per il calcolo dell'ϵ-GCD di polinomi; l'algoritmo unisce un basso costo computazionale a buone proprietà di stabilità.
- Proposte per sviluppi futuri:
  - sfruttare ulteriormente la struttura delle matrici risultanti (per es. nella fattorizzazione QR),
  - generalizzare al caso di molti polinomi, o di polinomi multivariati.

Bibliography I

D. A. Bini, P. Boito
A fast algorithm for approximate polynomial gcd based on structured matrix computations. *Preprint 1651, Department of Mathematics, University of Pisa (December 2006*).

I. Gohberg, T. Kailath, V. Olshevsky
Fast Gaussian elimination with partial pivoting for matrices with displacement structure.
Math. Comp., 64:1557–1576, 1995.

Bibliography II

G. Heinig
Inversion of generalized Cauchy matrices and other classes of structured matrices.
Linear Algebra in Signal Processing, IMA volumes in Mathematics and its Applications, 69:95-114, 1994.

A. Schönhage.
Quasi-GCD Computations.
J. Complexity, 1:118–137, 1985.

Z. Zeng.
The Approximate GCD Of Inexact Polynomials, *to appear*.