数理解析研究所講究録
第1976巻 2015年 8-16

区間演算を用いたQRGCD法
QRGCD method using interval arithmetic

沼畑 大
DAI NUMAHATA *
東京理科大学大学院 理学研究科
GRADUATE SCHOOL OF SCIENCE, TOKYO UNIVERSITY OF SCIENCE

Abstract

We use interval arithmetic to QRGCD method which obtains an approximate GCD by using QR decomposition. We describe problems while using interval arithmetic and solutions of the problems.

1 はじめに

多項式の最大公約因子 (GCD) を求める計算は数式処理において重要であり様々な方法があるが，多項式の係数に誤差がある問題において，通常のGCDを求める方法では一般には互いに素となる．このような多項式からも意味のある出力を得るために近似GCDという考え方が提案され，様々な研究がなされている．

本稿はその中でもQR分解を用いたQRGCD法と呼ばれる手法に焦点を当てる．QRGCD法はCorlessら [1] の研究があり改良が行われている [2]．本稿では区間演算を用いてQRGCD法の計算を行う．区間演算を用いたQRGCD法についてはKhungurnら [3] による安定化理論の立場からの研究があるが，本稿は別の立場で実装を試みる．

本稿では2章でQRGCD法と区間演算についての説明を行う．3章では区間演算を用いたQRGCD法についての説明を行う．4章では悪条件問題に対して，区間演算ではどのような問題があるのか説明する．5章ではまとめを行う。

2 背景と準備

2.1 QRGCD法

この節ではQRGCD法について概略を述べる．

定義 1 (Sylvester行列)
次数が m, n の2つの1変数実多項式

$$f(x) = f_m x^m + f_{m-1} x^{m-1} + \cdots + f_1 x + f_0$$
$$g(x) = g_n x^n + g_{n-1} x^{n-1} + \cdots + g_1 x + g_0$$

*nd0451@gmail.com

に対して $(m+n)\times(m+n)$ 行列

$$Syl(f,\ g)=\begin{pmatrix} f_m & f_{m-1} & \cdots & f_1 & f_0 & & & \\ & f_m & f_{m-1} & \cdots & f_1 & f_0 & & \\ & & \ddots & \ddots & & \ddots & \ddots & \\ & & & f_m & f_{m-1} & \cdots & f_1 & f_0 \\ g_n & g_{n-1} & \cdots & g_1 & g_0 & & & \\ & g_n & g_{n-1} & \cdots & g_1 & g_0 & & \\ & & \ddots & \ddots & & \ddots & \ddots & \\ & & & g_n & g_{n-1} & \cdots & g_1 & g_0 \end{pmatrix}$$

を Sylvester 行列とよぶ．f の係数が n 行並んだあと，g の係数が m 行並ぶ形になっている．

以下，f,g の係数ベクトルは 2 ノルムが 1 であると仮定する．
次の定理は QRGCD 法の基盤となる．

定理 2 (QR 分解による GCD の検出 [1])
$Syl(f,\ g)$ を $Syl(f,\ g)=QR$ ($Q\in\mathbb{R}^{(m+n)\times(m+n)}$ は直交行列，$R\in\mathbb{R}^{(m+n)\times(m+n)}$ は上三角行列)
と QR 分解すると，R の最後の 0 ベクトルではない行は $f(x)$ と $g(x)$ の GCD の係数ベクトルとなる．

QR 分解の手法には，Householder 変換，Givens 回転，その他工夫されたものなど様々あるが，本稿では Householder 変換を用いる．

例 1 (非自明な GCD をもつ例)
$f(x)=(x-0.1)(x+0.4)$, $g(x)=(x-0.1)(x-0.8)$ の GCD を求めることを考える．
$Syl(f,\ g)$ を精度 10 桁で QR 分解すると，

$$R=\begin{pmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & -0.7043635847 & 0.07043635855 \\ \cdots & \cdots & \cdots & 0 \end{pmatrix}.$$

よって $f(x)$, $g(x)$ の GCD は

$$-0.7043635847x+0.07043635855$$

と計算できる．

例 2 (互いに素な例)
$f(x)=(x-0.1)(x+0.4)$, $g(x)=(x-0.100001)(x-0.8)$ の GCD を求めることを考える．
$Syl(f,\ g)$ を精度 10 桁で QR 分解すると，

$$R=\begin{pmatrix} \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & -0.7043639250 & 0.07043681110 \\ \cdots & \cdots & \cdots & -1.772999996\times10^{-7} \end{pmatrix}.$$

この例は正確には互いに素だが $-1.772999996\times10^{-7}$ を微小な値とみなして，$-0.7043639250x+0.07043681110$ を近似的な GCD とみることができる．そこで，本稿では以下のように近似 GCD を定義する．

定義 3 (近似 GCD)
$Syl(f,\ g) = QR$ と QR 分解したときの R の行を係数ベクトルにもつ多項式 $d(x)$ で

$$f(x) + \Delta_f(x) = f_1(x)d(x),\ g(x) + \Delta_g(x) = g_1(x)d(x),$$
$$\deg(\Delta_f) \leq \deg(f),\ \deg(\Delta_g) \leq \deg(g),$$
$$||\Delta_f(x)||_2 \leq \varepsilon||f(x)||_2,\ ||\Delta_g(x)||_2 \leq \varepsilon||g(x)||_2$$

を満たす多項式 $f_1(x)$, $g_1(x)$, $\Delta_f(x)$, $\Delta_g(x)$ が存在するような最高次の $d(x)$ を許容度 ε の近似 GCD とよぶ．このように定義された近似 GCD を求める方法を QRGCD 法とよぶ．

注意 1
近似 GCD には様々な定義があり，この定義は QR 分解ありきの定義である．

2.2 区間演算

実数 a を近似するとき $a_0 \leq a \leq a_1$ と上下から挟んで区間と考え計算することで最終的な答の誤差の大きさを保証付で与える方法を区間演算とよぶ．a を近似して得られた値を $\langle a \rangle$，誤差の大きさを $\lfloor a \rceil$ とし，a を区間係数 $[[a]] = (\langle a \rangle,\ \lfloor a \rceil)$ で表現する．これは区間 $[\langle a \rangle - \lfloor a \rceil,\ \langle a \rangle + \lfloor a \rceil]$ を表す．
以下のように区間係数の演算を定義する．

定義 4 (区間演算)

$$fl_\tau:\ \text{精度}\tau\text{桁の浮動小数点で入力に最も近い値}$$
$$up_\tau:\ \text{精度}\tau\text{桁の切り上げ}$$
$$\varepsilon_\tau:\ \text{精度}\tau\text{桁の丸め誤差}$$

とすると，以下のように区間係数の演算が定義できる．

- 加法 $[[s]] = [[a]] + [[b]]$

$$\langle s \rangle = fl_\tau(\langle a \rangle + \langle b \rangle)$$
$$\lfloor s \rceil = up_\tau(\lfloor a \rceil + \lfloor b \rceil + \varepsilon_\tau(\langle s \rangle))$$

- 減法 $[[d]] = [[a]] - [[b]]$

$$\langle d \rangle = fl_\tau(\langle a \rangle - \langle b \rangle)$$
$$\lfloor d \rceil = up_\tau(\lfloor a \rceil + \lfloor b \rceil + \varepsilon_\tau(\langle d \rangle))$$

- 乗法 $[[p]] = [[a]][[b]]$

$$\langle p \rangle = fl_\tau(\langle a \rangle \langle b \rangle)$$
$$\lfloor p \rceil = up_\tau(\lfloor a \rceil \lfloor b \rceil + |\langle a \rangle| \lfloor b \rceil + \lfloor a \rceil |\langle b \rangle| + \varepsilon_\tau(\langle p \rangle))$$

- 逆数 $[[i]] = [[a]]^{-1}$ (区間が 0 を含まないとき)

$$\langle i \rangle = fl_\tau(1/\langle a \rangle)$$
$$\lfloor i \rceil = up_\tau\left(\frac{\lfloor a \rceil}{|\langle a \rangle|(|\langle a \rangle| - \lfloor a \rceil)} + \varepsilon_\tau(\langle i \rangle)\right)$$

- 平方根 $[[r]] = [[a]]^{1/2}$ (区間が $[0,\ \infty)$ の部分集合であるとき)

$$\langle r \rangle = fl_\tau(\langle a \rangle^{1/2})$$
$$\lfloor r \rceil = up_\tau\left(\frac{\langle a \rangle^{1/2}}{2}\left(\frac{\lfloor a \rceil}{\langle a \rangle - \lfloor a \rceil}\right) + \varepsilon_\tau(\langle r \rangle)\right)$$

例 3 (区間演算の加法)
精度 5 桁の数値 1.2345 と 6.7890 を区間演算で加算すると，それぞれ

$$1.2345 \Rightarrow (1.2345,\ 0.000050000),\ 6.7890 \Rightarrow (6.7890,\ 0.000050000)$$

と区間係数に変換して

$$(1.2345,\ 0.000050000) + (6.7890,\ 0.000050000) = (8.0235,\ 0.00015000)$$

となる．

区間演算のメリットとしては答えの精度が保証可能であることが挙げられるが，区間演算のデメリットとしては計算量の増加や，計算回数が増えることにより区間の幅が爆発的に大きくなってしまう危険性が挙げられる．

3 区間演算を用いた QRGCD 法

3.1 QRGCD 法に区間演算を用いるための設定

この章では QRGCD 法に区間演算を用いる方法を提案する．具体的には，$Syl(f,\ g)$ の各成分を区間係数に変換して QR 分解を行い，区間演算用に改良した条件分岐により非ゼロ行を検出する．

区間演算を用いない QRGCD 法では，QR 分解したときの R の行 $\vec{r}$ を 0 ベクトルと判定する際に条件式 $||\vec{r}||_2 < \varepsilon$ を用いる所を，条件 "$[[\,||\vec{r}||_2\,]] \subset (-\varepsilon,\ \varepsilon)$ ならば 0 ベクトル，$[[\,||\vec{r}||_2\,]] \subset (-\infty,\ -\varepsilon]$ または $[[\,||\vec{r}||_2\,]] \subset [\varepsilon,\ \infty)$ ならば非 0 ベクトル" で判定する．$(-\infty,\ -\varepsilon]$, $(-\varepsilon,\ \varepsilon)$, $[\varepsilon,\ \infty)$ のいずれの部分集合にもならない場合は部分集合になるまで精度を上げていけば，$(-\infty,\ -\varepsilon)$, $(-\varepsilon,\ \varepsilon]$, $[\varepsilon,\ \infty)$ のいずれかの部分集合となり，0 ベクトルかどうかの判定ができるとよいのだが残念ながらそうはならないことを以下の実験でみる．

3.2 実験

例 4 (通常の例)
$f(x) = (x - 0.1)(x + 0.4)$, $g(x) = (x - 0.100001)(x - 0.8)$ の GCD を求めることを考える．
$Syl(f,\ g)$ を精度 10 桁で QR 分解すると，R は

- 4 行目

  (4, 4) 成分: $(-1.772999996 \times 10^{-7},\ 1.779628970 \times 10^{-8})$

- 3 行目

  (3, 3) 成分: $(-0.7043639250,\ 1.720092168 \times 10^{-8})$

  (3, 4) 成分: $(0.07043681110,\ 1.847319472 \times 10^{-9})$

となる．適当な許容度 ε を設定すれば 3 行目が近似 GCD として検出される．

例 5 (悪条件の例)

$f(x) = (x - 0.005)(x + 5000)(x^3 + x^2 + x + 5)$, $g(x) = (x - 0.005)(x + 5000)(x^3 + 3x^2 + 7)$ の GCD を求める事を考える．

例 5 は以下の定理から悪条件とされる．

定理 5 (近似 GCD の小さい根 [1])

ω を，$f(x) + \Delta_f(x)$, $g(x) + \Delta_g(x)$ の $|\omega| < 1$ を満たす共通根とすれば，QRGCD 法で求めた近似 GCD の根となる．

定理 6 (近似 GCD の大きい根 [1])

ω を，$f(x) + \Delta_f(x)$, $g(x) + \Delta_g(x)$ の $|\omega| > 1$ を満たす共通根とすると，QRGCD 法で求めた近似 GCD の根となりにくい．

$Syl(f,\ g)$ を QR 分解すると，R の右下の要素は

精度 10 桁: $(0.004557404070,\ 0.00004382785861)$
精度 11 桁: $(-0.0039599037153,\ 0.0083175786725)$
精度 12 桁: $(-0.00230944770692,\ 0.00168273103492)$
精度 13 桁: $(-4.000000000000 \times 10^{-15},\ 0.0005867806202965)$
精度 14 桁: $(-0.0044368871990847,\ 0.0019013258859393)$
...

となる．区間演算を用いない通常の近似 GCD では単位円外の根が検出されないだけだが，区間演算ではさらに “精度を上げていけば，$(-\infty,\ -\varepsilon]$, $(-\varepsilon,\ \varepsilon)$, $[\varepsilon,\ \infty)$ のいずれかの部分集合となる” という期待が叶えられていない問題があることがこの例からわかる．自然に思われたこの期待が達成されなかった原因は，Householder 変換のアルゴリズムにある．

3.3 Householder 変換を用いた QR 分解

Householder 変換では，元の行列を 1 列ずつ処理して以下のように Q を作る．しかし，対角成分と対角成分より下の成分がすべて 0 ならばその列の変換を飛ばして次の列を処理するようになっている．

$$\overset{A_1}{\begin{pmatrix} * & * & * & * \\ * & * & * & * \\ * & * & * & * \\ * & * & * & * \end{pmatrix}} \Rightarrow \overset{A_2}{\begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \\ 0 & * & * & * \end{pmatrix}} \Rightarrow \overset{A_3}{\begin{pmatrix} * & * & * & * \\ 0 & * & * & * \\ 0 & 0 & * & * \\ 0 & 0 & * & * \end{pmatrix}}$$

具体的には以下のようなアルゴリズムである．

アルゴリズム 1 (Householder 変換)

$A_1 = Syl(f,\ g)$ に $j - 1$ 回 Householder 変換を施したものを A_j とする．

$$\sigma = \sqrt{a_{jj}^2 + \cdots + a_{dj}^2},\ A_j = \begin{pmatrix} a_{11} & & & & * \\ & \ddots & & & \\ & & a_{jj} & \cdots & a_{jd} \\ & & \vdots & & \vdots \\ 0 & & a_{dj} & \cdots & a_{dd} \end{pmatrix},\ v_j = \begin{pmatrix} a_{jj} \pm \sigma \\ a_{j,j+1} \\ \vdots \\ a_{dj} \end{pmatrix}.$$

$\sigma = 0$ のとき $A_{j+1} = A_j$, $\sigma \neq 0$ のとき $A_{j+1} = A_j - \dfrac{2v_j v_j^T}{v_j^T v_j}$.

区間演算では "$\sigma = 0$"は "$[[\sigma]]$ が 0 を含んでいる"となる．これでは普通の数値計算では Householder 変換を施すところを区間演算では飛ばす危険性があり，この条件分岐を正しく通過していないことが先述の期待を裏切る原因となったのである．区間演算では，区間が 0 を含んでしまうと区間幅とは関係なく逆数を求める計算ができないので，区間幅が小さくないのに 0 を含んでいるときは break して精度を上げる等の工夫する必要がある．

3.4 条件分岐を正しく通過するための条件

では，条件分岐を正しく通過するためにはどれくらい精度を上げればよいか．Khungurn ら [3] は正しい実行パスを通過する精度を与えているが，残念ながら現実的ではない．また，一般に近似 GCD を求める問題では f, g は正確には互いに素であることが多く，すなわち $\sigma = 0$ の TRUE はまず正確なパスではない．したがって，余りこだわらなくてもよいと言える問題だが，念のためうまく判定できるような実装を施した．これは完璧に正しく条件分岐を通過することができるといえるものではないが，通常の数値計算でも同じような完全ではない工夫がなされているので実際上は問題ないと思われる．

アルゴリズム 2 (0 判定)
if v のすべての要素に 0 が含まれている **then**
begin
　if σ が適当な区間 $(-\varepsilon,\ \varepsilon)$ に含まれている or v の第 1 成分 $> M$ （M は適当な十分大きな値）
　then $A_{j+1} = A_j$
end else break %精度を上げてやり直し

通常の数値計算では $\sigma < \varepsilon =$ マシンイプシロンだが，本稿の区間演算では指数部は多倍長を仮定しているので，$(-\varepsilon,\ \varepsilon)$ に Householder 変換において 0 とみなした要素を採用している．

3.5 アルゴリズム

区間演算を用いた QRGCD 法のアルゴリズムを以下のように提案する．

アルゴリズム 3 (区間演算を用いた QRGCD 法)
Input: 1 変数実多項式 $f(x)$, $g(x)$, 許容誤差 ε, 初期精度 m
Output: f, g の近似 GCD
Step 1. 精度 m 桁 で $Syl(f,\ g) = QR$ と QR 分解
　　許容誤差の範囲でできなければ $m = m + 1$ として繰り返す
Step 2. 近似 GCD の定義を満たす多項式の係数ベクトルを R の下の行からアルゴリズム 2 を用いて
　　探索し見つかった中で最高次のものを近似 GCD として出力
　　1 つも候補がない場合は $m = m + 1$ として Step 1 へ

ここで，探索の際，除算の代わりに $LC(g)f - LC(f)g$, $\varepsilon := LC(g)\varepsilon$ とする先頭項打ち消しの操作をする ($LC(f)$ は f の先頭係数を表す)．

4　悪条件問題に対しての区間演算

例 5 でみたような悪条件問題について，区間演算を用いると区間幅が劇的に大きくなる要因として桁落ちが挙げられる．Householder 変換のどこで桁落ちが起きるかを観察すると，以下のようになる．
× は桁落ち，△ は誤差伝播とする．

- $j=1$

$$\begin{pmatrix}
* & * & * & \cdots & * & * & & & & \\
 & * & * & * & \cdots & * & * & & & \\
 & & * & * & * & \cdots & * & * & & \\
 & & & \ddots & \ddots & & \ddots & & \ddots & \\
 & & & & * & * & * & \cdots & * & * \\
* & * & * & \cdots & * & * & & & & \\
 & * & * & * & \cdots & * & * & & & \\
 & & * & * & * & \cdots & * & * & & \\
 & & & \ddots & \ddots & & \ddots & & \ddots & \\
 & & & & * & * & * & \cdots & * & *
\end{pmatrix}$$

- $j=2$

$$\begin{pmatrix}
* & * & * & \cdots & * & * & & & & \\
 & * & * & * & \cdots & * & * & & & \\
 & & * & * & * & \cdots & * & * & & \\
 & & & \ddots & \ddots & & \ddots & & \ddots & \\
 & & & & * & * & * & \cdots & * & * \\
0 & \times & * & \cdots & * & * & & & & \\
 & * & * & * & \cdots & * & * & & & \\
 & & * & * & * & \cdots & * & * & & \\
 & & & \ddots & \ddots & & \ddots & & \ddots & \\
 & & & & * & * & * & \cdots & * & *
\end{pmatrix}$$

- $j=3$

$$\begin{pmatrix}
* & * & * & \cdots & * & * & & & & \\
 & * & * & * & \cdots & * & * & & & \\
 & & * & * & * & \cdots & * & * & & \\
 & & & \ddots & \ddots & & \ddots & & \ddots & \\
 & & & & * & * & * & \cdots & * & * \\
0 & 0 & \times & \triangle & \cdots & \triangle & \triangle & \triangle & \triangle & \triangle \\
 & 0 & \times & \triangle & \cdots & \triangle & \triangle & \triangle & \triangle & \triangle \\
 & & * & * & * & \cdots & * & * & & \\
 & & & \ddots & \ddots & & \ddots & & \ddots & \\
 & & & & * & * & * & \cdots & * & *
\end{pmatrix}$$

- $j=4$

$$\begin{pmatrix}
* & * & * & \cdots & * & * & & & & \\
 & * & * & * & \cdots & * & * & & & \\
 & & * & * & * & \cdots & * & * & & \\
 & & & \ddots & \ddots & & \ddots & & \ddots & \\
 & & & & * & * & * & \cdots & * & * \\
0 & 0 & 0 & \times & \triangle & \cdots & \triangle & \triangle & \triangle & \triangle \\
 & 0 & 0 & \times & \triangle & \cdots & \triangle & \triangle & \triangle & \triangle \\
 & & 0 & \times & \triangle & \cdots & \triangle & \triangle & \triangle & \triangle \\
 & & & \ddots & \ddots & & \ddots & & \ddots & \\
 & & & & * & * & * & \cdots & * & *
\end{pmatrix}$$

なぜこのような桁落ちが起きるかというと α を絶対値が非常に大きい実数として

$$f=(x-\alpha)f' \text{とすると } f_{m-1}=-f_m(\alpha+\cdots)\approx -f_m\alpha,$$
$$g=(x-\alpha)g' \text{とすると } g_{n-1}=-g_n(\alpha+\cdots)\approx -g_n\alpha$$

となり，近似式 $\dfrac{f_m}{f_{m-1}}\approx\dfrac{g_n}{g_{n-1}}$ が成立するからである．初回の Householder 変換の桁落ちはこの近似式による．この近似式を用いると

$$Syl(f,\ g)=\begin{pmatrix}
f_m & f_{m-1} & \cdots & f_1 & f_0 & & & \\
 & f_m & f_{m-1} & \cdots & f_1 & f_0 & & \\
 & & \ddots & \ddots & & \ddots & \ddots & \\
 & & & f_m & f_{m-1} & \cdots & f_1 & f_0 \\
g_n & g_{n-1} & \cdots & g_1 & g_0 & & & \\
 & g_n & g_{n-1} & \cdots & g_1 & g_0 & & \\
 & & \ddots & \ddots & & \ddots & \ddots & \\
 & & & g_n & g_{n-1} & \cdots & g_1 & g_0
\end{pmatrix}$$

は基本変形で $(n+1,\ 2)$ 成分が桁落ちすることから明らかだが，実際に計算してみると

$$\begin{aligned}
g_{n-1}-\frac{2}{v^Tv}(v^T\mathbf{a}_2)g_n &= g_{n-1}-\frac{g_n((f_m+\sqrt{f_m^2+g_n^2})f_{m-1}+g_ng_{n-1})}{f_m^2+g_n^2+f_n\sqrt{f_m^2+g_n^2}} \\
&\approx g_{n-1}-\frac{f_n\sqrt{f_m^2+g_n^2}g_{n-1}f_m+g_n^2g_{n-1}}{f_m^2+g_n^2+f_n\sqrt{f_m^2+g_n^2}} \\
&\quad \left(\text{近似式}\frac{f_m}{f_{m-1}}\approx\frac{g_n}{g_{n-1}}\iff f_mg_{n-1}=f_{m-1}g_n \text{ を使った}\right) \\
&= 0.
\end{aligned}$$

次の桁落ちは近似式 $\dfrac{f_m}{f_{m-1}}\approx\dfrac{g_n}{g_{n-1}}\approx\dfrac{h_{n-1}}{h_{n-2}}$ による．実際 $h=f-ag=(x-\alpha)(f'-ag')$ とすると，

$f' - ag'$ が α より絶対値の大きい根をもつことはほとんどないといえるので，上の近似式が成り立つ．

$$Syl(f,\ g) = \begin{pmatrix} * & * & \cdots & * & * & & & \\ & f_m & f_{m-1} & \cdots & f_1 & f_0 & & \\ & & \ddots & \ddots & & \ddots & \ddots & \\ & & & f_m & f_{m-1} & \cdots & f_1 & f_0 \\ 0 & h_{m-1} & h_{m-2} & \cdots & h_0 & & & \\ & g_n & g_{n-1} & \cdots & g_1 & g_0 & & \\ & & \ddots & \ddots & & \ddots & \ddots & \\ & & & g_n & g_{n-1} & \cdots & g_1 & g_0 \end{pmatrix}$$

このような桁落ちが次々と起こることから，区間演算においてこの種の悪条件問題は区間の幅が爆発的に大きくなり最終的に収束しにくい．

一方，絶対値が非常に大きい根を含まなければ v の第 1 要素の絶対値が大きくなり，この種の桁落ちは起きにくい．

5 まとめ

本稿では区間演算を用いた QRGCD 法の実装をし，区間演算をする上で必要な様々な工夫を施した．今後の課題として，より効率のよいアルゴリズムの設計や，他の工夫された QR 分解，他のアルゴリズムにおける同様の手法での区間演算の利用が挙げられる．

参 考 文 献

[1] R. Corless, S. Watt, and L. Zhi. QR factoring to compute the GCD of univariate approximate polynomials. *IEEE Trans. Signal Processing*, vol. 52, No. 12, 2004, pp. 3394–3402.

[2] 増井貴明, 長坂耕作. SNAP package and Improved QRGCD algorithm. 数理解析研究所講究録, 第 1843 巻, 2013 年, pp. 101–113.

[3] P. Khungurn, H. Sekigawa, K. Shirayanagi. Minimum converging precision of the QR-factorization algorithm for real polynomial. *Proc. ISSAC 2007*, 2007, pp. 227–234.