دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

دانشکده ریاضی و علوم کامپیوتر

پایان‌نامه کارشناسی‌ارشد

گرایش ریاضی

یافتن ریشه‌های چندجمله‌ای با ماتریس همراه آن

نگارش

آترینحجت

بهمن ۱۴۰۲

فهرست مطالب

عنوان صفحه

فهرست تصاویر

شکل صفحه

فهرست جداول

جدول صفحه

فهرست نمادها

نماد	مفهوم
$p(x)$	چند جمله‌ای بر پایه‌ی x

فصل اول

شرح مسئله

۱-۱ مقدمه

هدف این پروژه، بررسی روش یافتن ریشه‌های یک چند جمله‌ای با استفاده از ماتریس همراه [۱] آن و گسترش دادن این روش برای مسائل دیگر است. در ابتدا، روش‌های مختلف یافتن ریشه‌های چندجمله‌ای را بررسی کرده و سپس روش ماترسی همراه را معرفی و اثبات می‌کنیم. در این راستا، خواص این ماتریس و ارتباط ریشه‌های چندجمله‌ای با مقادیر ویژه‌ی ماتریس را تحلیل می‌کنیم.

در ادامه، به روش‌های محاسباتی این ماتریس و مقایسه‌ی آن با دیگر روش‌های می‌پردازیم. دقت و صحت آن را در محاسبات مختلف سنجیده و نقاط قوت و ضعف آن را بررسی می‌کنیم و کاربرد عملی آن را پیاده سازی می‌کنیم.

با توجه به بررسی نقاط قوت این روش، چند نمونه‌ی عملی از کاربردهای آن را می‌پردازیم و استفاده‌های آن در حوزه‌های مختلف را شرح می‌دهیم.

در آخر، به کاربرد ماتریس همراه در حوزه‌های دیگر می‌پردازیم، از روش مشابهی، برای حل معادلات دیفرانسیل و معادلات مثلثاتی استفاده می‌کنیم و کاربرد های گسترده‌تر این روش رو بررسی می‌کنیم.

برای مشاهده‌ی کدهای استفاده شده در این گزارش می‌توانید به اینجا مراجعه کنید.

۱-۲ شرح مسئله

پیدا کردن ریشه‌های یک چند جمله‌ای، از مسائل پرکاربرد در حوزه‌های مختلف علم است. چندجمله‌ای‌های بستری منعطف برای مدل کردن رابطه‌ی بین متغیرها، توصیف فرایندهای پیچیده و پیش‌بینی نتایج فراهم می‌کنند. بررسی رفتار و ریشه‌های چندجمله‌ای ها فهم عمیقی از رفتار سیستم‌های و رابطه‌ی میان عوامل مختلف ارائه می‌دهد.

در علوم کامپیوتر و CAD ، می‌توان از ریشه‌های چند جمله‌ای برای Curve Fitting استفاده کرد که لازمه‌ی واقعی‌دیده شدن منحنی‌ها در گرافیک است. به منظور تصحیح خطا [۲] در رمزنگاری [۳] برای encode و decode کردن اطلاعات از چندجمله‌ای ها استفاده می‌شود تا از صحت اطلاعات اطمینان برقرار شود و خطاهای بوجود آمده در زخیره‌سازی و جابجایی اطلاعات تصحیح شود [۳] . همچنین بساری از الگوریتم‌های کدگذاری، از چندجمله‌ای‌ها در همنهشتی اعداد اول استفاده می‌کنند و پیدا کردن ریشه‌های این چندجمله‌ای ها میتواند به شکستن این کدگذاری‌ها منجر شود. از دیگر کاربردهای آن، می‌توان به مدل‌های اقتصادی، طراحی فیلتر در Signal Processing ، رگرسیون [۴] در آمار و... اشاره کرد.

یک چند جمله‌ای بطور کلی، به فرم:

---

[۱] Companion Matrix

[۲] Error Correcting

[۳] Cryptography

[۴] Regression

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \qquad (۱-۱)$$

می‌باشد. در این گزارش برای ساده‌سازی، از فرم زیر استفاده می‌کنیم:

$$p(x) = x^n + \frac{a_{n-1}}{a_n}x^{n-1} + \cdots + \frac{a_1}{a_n}x + \frac{a_0}{a_n} = x^n + b_{n-1}x^{n-1} + \cdots + b_1 x + b_0 \qquad (۲-۱)$$

فرض کنید $x_1, x_2, \cdots, x_n$ ریشه های چندجمله ای باشند، یعنی:

$$\forall 1 \leq i \leq n : p(x_i) = 0 \qquad (۳-۱)$$

$$p(x) = (x - x_1)(x - x_2)\cdots(x - x_n) \qquad (۴-۱)$$

هدف پیدا کردن x_i هاست. می‌دانیم هر چندجمله‌آی از درجه‌ی n ، دقیقا n ریشه (حقیقی یا موهومی) دارد در ابتدا می‌خواهیم چند روش کلی برای حل این مسئله را به همراه پیچیدگی محاثباتی آنها بررسی کنیم.

۱-۳ روش‌های بدست آوردن ریشه‌های چندجمله‌ای

۱-۳-۱ روش نیوتون

روش نیوتون [۵] ، یکی از روش‌های پرکاربرد برای یافتن یک ریشه از چند جمله‌ای‌ست. در این روش، با شروع از x_0 دلخواه، در هر گام

$$x_{n+1} = x_n - p(x_n)/p'(x_n) \qquad (۵-۱)$$

[۵] Newton's Method

$$p'(x) = nx^{n-1} + (n-1)a_{n-1}x^{n-2} + \cdots + a_1 \quad (۱-۶)$$

این روش معمولا در $O(n^2)$ مرحله ریشه‌ای برای چندجمله‌ای پیدا می‌کند. اما اگر چندجمله‌ای ریشه‌ی حقیقی نداشته باشد و x_0 حقیقی باشد، الگوریتم هیچ‌گاه به یک ریشه‌ی چند جمله‌ای نمیرسد. اگر x_0 بزرگ تر از بزرگترین ریشه چندجمله‌ای باشد، این الگوریتم در $O(n^2)$ مرحله به پایان می‌رسد و بزرگترین ریشه‌ی چندجمله‌ای را پیادا می‌کند.

هر مرحله ای این الگوریتم، نیازمند به‌دست آوردن مقدار چندجمله‌ای و مشتق آن است که به پیچیدگی محاثباتی $O(n^3 \log n)$ یا $O(n^4)$ منجر می‌شود که با استفاده از Preprocessing یا Dynamic Programming می‌توان آن را به $O(n^3)$ کاهش داد.

۱-۳-۲ روش هرنر

در این روش، چند جمله‌ای p را به فرم زیر بازنویسی می‌کنیم:

$$p(x) = a_0 + x(a_1 + x(a_2 + x(\cdots + x(a_{n-1} + xa_n)\cdots))) \quad (۱-۷)$$

برای محاسبه‌ی این عبارت، میتوان از سری زیر استفاده کرد

$$\begin{aligned} b_n &= a_n \\ b_{n-1} &= a_{n-1} + b_n x_0 \\ &\vdots \\ b_1 &= a_1 + b_2 x_0 \\ b_0 &= a_0 + b_1 x_0 \end{aligned} \quad (۱-۸)$$

که در آن $p(x_0) = b_0$.

فرض کنید چندجمله‌ای دارای ریشه‌های

$$z_n < z_{n-1} < \cdots < z_1 \quad (۱-۹)$$

باشد. روش هرنر [۶] در ابتدا با یک حدس $z_1 < x_0$ شروع می‌کند. سپس با روش نیوتون، ریشه z_1

[۶] Horner's Method

را می‌یابیم. سپس $p(x)/(x-z_1)$ را حساب کرده، و با شروع از z_1 همین روند را برای z_2 تا z_n تکرار می‌کنیم. حال می‌توان نشان داد

$$p(x) = (b_1 + b_2x + b_3x^2 + \cdots + b_{n-1}x^{n-2} + b_nx^{n-1})(x - x_0) + b_0 \tag{۱۰-۱}$$

که برای $x_0 = z_1$

$$b_0 = p(x_0) = 0 \implies p(x)/(x - x_0) = b_1 + b_2x + b_3x^2 + \cdots + b_{n-1}x^{n-2} + b_nx^{n-1} \tag{۱۱-۱}$$

۱-۳-۳ روش دورند-کرنر

فرض کنید چند جمله‌ای درجه ۳ پایین را داریم: [۷]

$$p(x) = x^3 + a_2x^2 + a_1x + a_0 \tag{۱۲-۱}$$

اگر A ، B و C ریشه‌های این چند جمله‌ای باشند، داریم:

$$p(x) = (x - A)(x - B)(x - C) \tag{۱۳-۱}$$

$$A = x - \frac{p(x)}{(x - B)(x - C)} \tag{۱۴-۱}$$

برای $x_0 \neq B, C$

$$x_1 = x_0 - \frac{p(x_0)}{(x_0 - B)(x_0 - C)} \tag{۱۵-۱}$$

[۷] Durand-Kerner method

این عملیات در یک مرحله P را به دست می‌آورد. حال با حدس‌های اولیه a_0, b_0, c_0 می‌توانیم این محاسبات را تکرار کنیم تا مقادیر A, B, C به دست آیند. فقط حدس های اولیه باید غیر حقیقی باشند و ریشه ۱ نباشند

$$\begin{aligned}
a_{k+1} &= a_k - \frac{p(a_k)}{(a_k - b_k)(a_k - c_k)} \\
b_{k+1} &= b_k - \frac{p(b_k)}{(b_k - a_k)(b_k - c_k)} \\
c_{k+1} &= c_k - \frac{p(c_k)}{(c_k - a_k)(c_k - b_k)}
\end{aligned} \tag{۱-۱۶}$$

این روش‌را برای چند جمله‌ای با درجه‌ی دلخواه می‌توان گسترش داد.

فصل دوم

روش ماتریس همراه

در این فصل، روش ماتریس همراه را معرفی می‌کنیم. این روش، قابل اعتمار ترین روش برای محاسبه‌ی ریشه‌های تکراری یک چند جمله‌ای ست. هردو روش بررسی شده در بالا، با فرض عدم وجود ریشه‌های تکراری کار می‌کنند. محاسبه‌ی این ریشه‌ها در عملیات‌های کامپیوتر معمولا خطای زیادی دارد. از این رو پیدا کردن روشی که بتواند این ریشه‌ها را نیز با دقت خوبی محاسبه کند بسیار حائز اهمیت است.

۲-۱ بررسی روش ماتریس همراه

فرض کنید چند جمله‌ای p به فرم

$$p(x) = x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0 \tag{۲-۱}$$

داده شده است. ماتریس همراه آن را به شکل زیر تعریف می‌کنیم:

$$C = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & 0 & \cdots & 0 & -a_2 \\ 0 & 0 & 1 & \cdots & 0 & -a_3 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -a_{n-1} \end{bmatrix} \tag{۲-۲}$$

در این روش، مقادری ویژه‌ی ماتریس C همان ریشه‌های چندجمله‌ای p خواهد بود.

این روش سریع‌ترین یا بهینه ترین روش برای محاسبه‌ی ریشه‌های چند جمله‌ای نیست، از آنجایی که نیاز به $O(n^2)$ حافظه و $O(n^3)$ محاسبه دارد، در حالی که یک الگوریتم بهینه برای به دست آوردن ریشه‌های چندجمله‌ای میتواند با حافظه $O(n)$ و پیچیدگی محاسباتی $O(n^2)$ کار کند. از طرفی، هرچه تعداد محاسبات بیشتر باشد، میزان خطای این روش نیز بیشتر خواهد بود [۶] .

در محاسبه‌ی مقادیر ویژه‌ی یک ماتریس، میزان خطا در کامپیوتر حداقل از اردر $O(\epsilon||A||_F)$ که ϵ دقت ماشین و $||A||_F$ نرم فروبنیوس ماتریس است [۷] . برای کاهش این خطا، می‌توان از تغییرات قطری [۱] استفاده کرد بطوری‌که نرم A کاهش یابد [۱] [۵] [۷] .

پس از نرمال کردن ماتریس، می‌توان از الگوریتم QR برای به دست آوردن مقادیر ویژه‌ی ماتریس استفاده کرد. در این الگوریتم، در مرحله k ام، ابتدا تجزیه‌ی QR برای ماتریس A_k به فرم $A_k = Q_k R_k$ را به دست می‌آوریم که $A_0 = A$. سپس در هر مرحله قرار می‌دهیم $A_{k+1} = R_k Q_k$. با توجه به معدله‌ی زیر، می‌توان دید که در هر مرحله، A_k مشابه A است.

[۱] Diagonal similarity transformations

$$A_{k+1} = R_kQ_k = Q_k^{-1}Q_kR_kQ_k = Q_k^{-1}AQ_k = Q_k^TAQ_k \qquad (۳-۲)$$

پس از تعدادی مرحله، ماتریس A_k به ماتریسی مثلثی تبدیل می‌شود که به آن شکل شور [۲] A گفته می‌شود. مقادیر ویژه‌ی ماتریس‌های مثلثی، برابر مقادیر روی قطر آنهاست. با توجه به اینکه A و A_k مشابه‌اند، مقادیر ویژه‌ی A نیز به دست آمده است. ممکن است شرایطی به وجود آید که A_k مثلثی نشود [۴] .

۲-۲ اثبات روش ماتریس همراه

به منظور اثبات درستی این روش، کافیست نشان دهیم که چندجمله‌ای مشخصه‌ی ماتریس C به فرم $det(Ix - C)$ برابر چندجمله‌ای $p(x)$ است.

برای اثبات صحت این روش، از استقرا استفاده می‌کنیم:

اثبات. پایه: $(n = 1)$

$$A = [a_0], det(Ix - A) = x - a_0$$

گام:

$$Ix - C = \begin{bmatrix} x & 0 & 0 & \cdots & 0 & a_0 \\ -1 & x & 0 & \cdots & 0 & a_1 \\ 0 & -1 & x & \cdots & 0 & a_2 \\ 0 & 0 & -1 & \cdots & 0 & a_3 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & -1 & x + a_{n-1} \end{bmatrix} \qquad (۴-۲)$$

[۲] Schor form of A

(۲-۵)

$$det(Ix-C)=\begin{vmatrix} x & 0 & 0 & \cdots & 0 & a_0 \\ -1 & x & 0 & \cdots & 0 & a_1 \\ 0 & -1 & x & \cdots & 0 & a_2 \\ 0 & 0 & -1 & \cdots & 0 & a_3 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & -1 & x+a_{n-1} \end{vmatrix}$$

$$=x\begin{vmatrix} x & 0 & \cdots & 0 & a_1 \\ -1 & x & \cdots & 0 & a_2 \\ 0 & -1 & \cdots & 0 & a_3 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & \cdots & -1 & x+a_{n-1} \end{vmatrix} - (-1)^{n+1}a_0 \begin{vmatrix} x & 0 & \cdots & 0 \\ -1 & x & \cdots & 0 \\ 0 & -1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & -1 \end{vmatrix}$$

طبق فرض استقرا داریم:

$$\begin{vmatrix} x & 0 & \cdots & 0 & a_1 \\ -1 & x & \cdots & 0 & a_2 \\ 0 & -1 & \cdots & 0 & a_3 \\ \vdots & & & \ddots & \vdots \\ 0 & 0 & \cdots & -1 & x+a_{n-1} \end{vmatrix} = x^{n-1}+a_{n-1}x^{n-2}+\cdots+a_2x+a_1 \qquad (۲-۶)$$

و دترمینان ماتریس دوم به دلیل مثلثی بودن، برابر با حاصل ضرب مقادیر روی قطر آن است

$$\begin{aligned} det(Ix-C) &= x(x^{n-1}+a_{n-1}x^{n-2}+\cdots+a_2x+a_1)+(-1)^{n+1}a_0(-1)^{n-1} \\ &= x^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0 \end{aligned} \qquad (۲-۷)$$

□

۲-۳ محدود کردن بازه‌ی ریشه‌های معادله

همانطور که در بخش ۱ دیدیم، روش‌هایی مانند روش نیوتون نیازمند وجود یک تخمین از محدوده‌ی ریشه‌هاست. برای مثال، روش نیوتون نیاز داشت که حداکثر مقدار ریشه‌های چندجمله‌ای را داشته باشد، تا حدس اولیه‌اش بزرگ تر از بزرگترین ریشه‌ی چندجمله‌ای باشد. در این بخش می‌خواهیم استفاده‌ی ماتریس همراه را در تخمین مقادیر ریشه‌ها بررسی کنیم.

در فصل ۵ درس، دیدیم که می‌توان از حلقه‌های گرشگروین [۳] استفاده کرد تا محدوده‌ی مقادیر ویژه‌ی یک ماتریس را به دست آورد. شعاع هر یک از این n حلقه از رابطه‌ی

$$r_i = \sum_{\substack{j=1 \\ i \neq j}}^{n} |a_{ij}| \tag{۲-۸}$$

هر یک از مقادیر ویژه‌ی ماتریس A باید حداقل در یکی از نامساوی‌های زیر صدق کند:

$$|\lambda - a_{ii}| \leq r_i \tag{۲-۹}$$

از آنجایی که

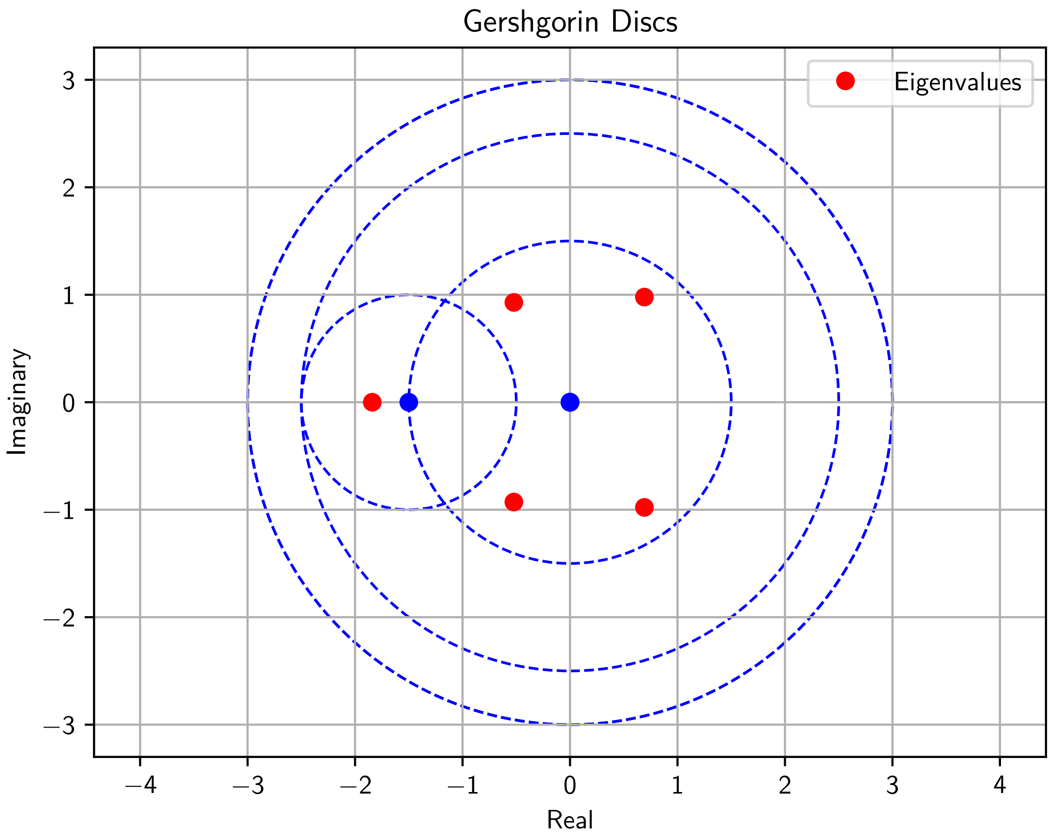
$$C = \begin{bmatrix} 0 & 0 & 0 & \cdots & 0 & -a_0 \\ 1 & 0 & 0 & \cdots & 0 & -a_1 \\ 0 & 1 & 0 & \cdots & 0 & -a_2 \\ 0 & 0 & 1 & \cdots & 0 & -a_3 \\ \vdots & & & \ddots & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & -a_{n-1} \end{bmatrix} \tag{۲-۱۰}$$

داریم

$$\begin{cases} |\lambda| \leq a_i & 0 \leq i < n-1 \text{or} \\ |\lambda + a_{n-1}| \leq 1 & \text{else.} \end{cases} \tag{۲-۱۱}$$

یعنی یک دیسک با شعاع 1 در مرکز a_{n-1} و $n-2$ دیسک با مرکز 0 قرار می‌گیرند. پس می‌توان

[۳] Gershgorin



شکل ۲-۱: Gershgorin Discs for a Companion Matrix

گفت که ریشه‌های یک چندجمله‌ای دی یکی از دو بازه‌ی زیر اند

$$\begin{cases} |\lambda| \leq \max_{0\leq i<n-1} a_i & \text{or} \\ |\lambda + a_{n-1}| \leq 1 & \text{else.} \end{cases} \tag{۲-۱۲}$$

۲-۴ پیدا کردن کوچکترین یا بزرگترین ریشه‌ی یک چندجمله‌ای با کمک روش توانی

می‌دانیم یک روش برای محاسبه‌ی یک مقدار ویژه‌ی ماتریس، استفاده از روش توانی است. در این ورش، میتوان بصورت تکراری با شروع از یک بردار دلخواه، بزرگترین مقدار ویژه‌ی یک ماتریس را محاسبه‌کرد.

در این روش با شروع از $V^{(0)}$ در هر مرحله

$$V^{(i)} = AV^{(i-1)} \tag{۲-۱۳}$$

برای کاهش خطای محاسبه می‌توان قرار داد:

$$\begin{aligned} V^{(i)} &= A\tilde{V}^{(i-1)} \\ \tilde{V}^{(i)} &= V^{(i)}/||V^{(i)}||_\infty \end{aligned} \tag{۲-۱۴}$$

بطوری که

$$\lambda \approx \frac{\tilde{V}_j^{(i)}}{\tilde{V}_j^{(i-1)}} \tag{۲-۱۵}$$

برای محاسبه‌ی کوچکترین مقدار ویژه، کافیست بجای A قرار دهیم A^{-1} . در این صورت داریم

$$\begin{aligned} AV^{(i)} &= \tilde{V}^{(i-1)} \\ \tilde{V}^{(i)} &= V^{(i)}/||V^{(i)}||_\infty \end{aligned} \tag{۲-۱۶}$$

که نیازمند حل معادله $AV^{(i)} = \tilde{V}^{(i-1)}$ خواهد بود.

حال کافیست ماتریس همراه چندجمله‌ای را در این الگوریتم قرار بدهیم تا بزرگترین یا کوچکترین ریشه‌ی‌آن را بیابیم. واضح است که پیچیدگی پردازشی این روش، $O(kn^2)$ خواهد بود که برای n های بزرگ بهینه تر از روش نیوتون است.

فصل سوم

پیاده‌سازی و پیچیدگی محاسباتی

۳–۱ پیاده‌سازی روش ماتریس همراه در پایتون

در این بخش می‌خواهیم به نحوه‌ی پیاده سازی این الگوریتم در پایتون بپردازیم. فرض کنید چند جمله‌ای دلخواه $p(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0$ داده شده است. در مرحله‌ی اول، باید تمامی ضرایب چند جمله ای را بر a_n تقسیم کنیم، تا این چند جمله‌ای به فرم مورد نظر ما در آید. برای ورودی گرفتی یک چند جمله‌ای، کافیست یک آرایه $n+1$ تایی داشته باشیم که اندیس i ام آن متناظر با a_{n-i} است.

```
def get_formated_poly(poly):
    """
    Return the polynomial such that the coefficient of the maximum power of x
    is always 1
    """
    ret =poly /poly[0]
    return ret
```

در مرحله‌ی بعد، باید ماتریس همراه این چندجمله‌ای را بسازیم:

```
def get_companion_matrix(poly):
    """
    Calculate the companion matrix for a normalized polynomial
    """
    n =len(poly)
    cmat =np.zeros((n -1, n -1))
    cmat[:, n -2] =-poly[1:]
    cmat[np.arange(1, n -1), np.arange(0, n -2)] =1

    return cmat
```

در آخر، باید مقادیر ویژه‌ی این ماتریس را حساب کنیم. به این منظور می‌توانیم از بخش جبر خطی کتابخوانه numpy استفاده کنیم یا الگوریتم QR را پیاده سازی کنیم:

```
poly =np.array([2, 3, 1, 4, 3, 6])
norm_poly =get_formated_poly(poly)


eigenvalues, _ =np.linalg.eig(matrix)
```

این روش به طور کلی خطای پایینی دارد و اگرچه ممکن است با ورودی‌های خواص، میزان خطا افزایش یابد، این میزان در طول محاسبات برای QR و دترمینان ماتریس نسبتا ثابت می‌مانند [۲]

۳-۲ محاسبه حدود ریشه‌ها

در این بخش، حلقه‌های گرشگورین رو برای ماتریس‌همراه محاسبه‌می‌کنیم. به این منظور، پس از نرمال کردن چندجمله‌ای و به دست آوردن ماتریس همراه، کافیست بزرگترین مقدار بین $|a_0|$ تا $|a_{n-2}|$ را به دست آورده(فرض کنید a_k بزرگترین است)، و با $a_{n-1} \pm 1$ مقایسه کنیم . ریشه‌های معادله در یکی از

$$\begin{cases} |\lambda| \leq a_k \\ |\lambda + a_{n-1}| \leq 1 \end{cases} \tag{۳-۱}$$

قرار خواهند داشت.

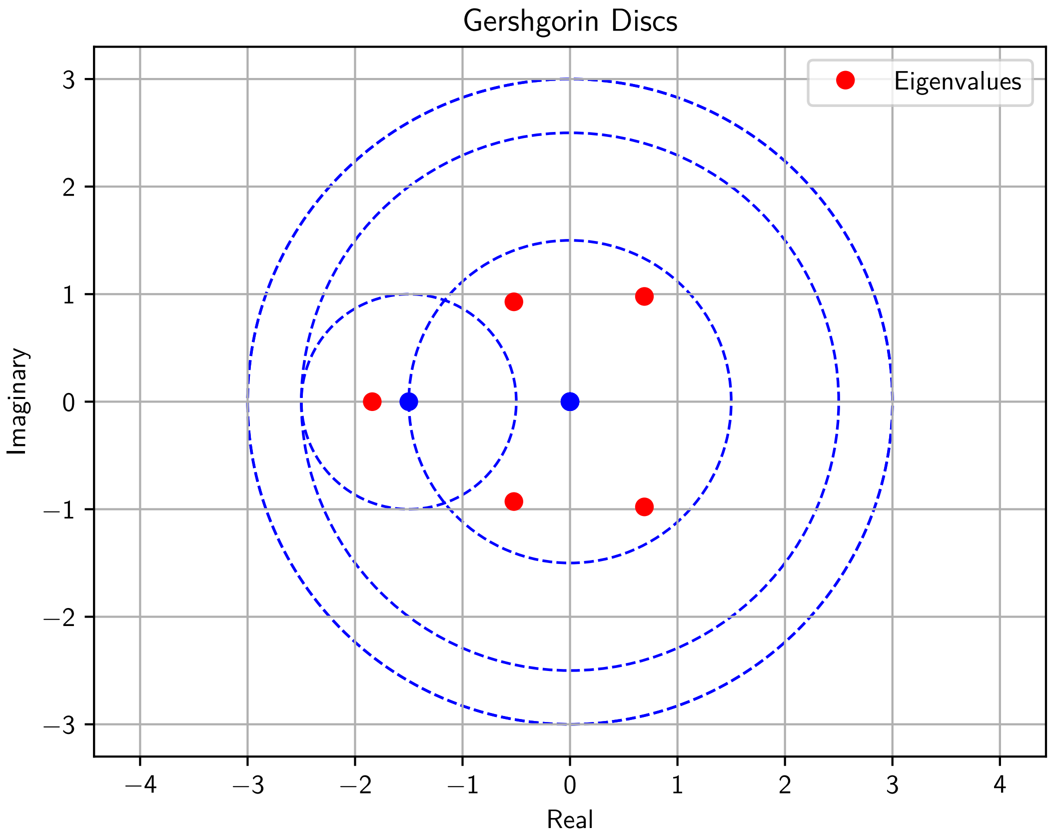
```
bounds =[
  (-np.max(np.abs(norm_poly[2:])), +np.max(np.abs(norm_poly[2:]))),
  (-norm_poly[1] -1, -norm_poly[1] +1),
]
```

```
import os
import numpy as np
import matplotlib
import matplotlib.pyplot as plt



def plot_gershgorin_discs(matrix):
    n =len(matrix)
    eigenvalues, _ =np.linalg.eig(matrix)

    fig, ax =plt.subplots()
```



شکل ۳-۱: Gershgorin Discs for a Companion Matrix

```python
ax.set_aspect('equal', adjustable='datalim')

for i in range(n):
    disc_center =matrix[i, i]
    disc_radius =np.sum(np.abs(matrix[i, :])) -np.abs(matrix[i, i])

    disc =plt.Circle((disc_center, 0), disc_radius, fill=False, color='b',
                                        linestyle='dashed')
    ax.add_patch(disc)
    ax.plot(disc_center, 0, 'bo') # Plot the center of the disc

ax.set_title('Gershgorin Discs')
ax.grid(True)
plt.xlabel('Real')
plt.ylabel('Imaginary')

ax.plot(np.real(eigenvalues), np.imag(eigenvalues), 'ro',
                                    label='Eigenvalues')
plt.legend()

plt.savefig(os.path.join("../output", "gershgorin.jpg"))
matplotlib.rcParams.update({
    "pgf.texsystem": "xelatex",
    'text.usetex': True,
    'pgf.rcfonts': False,
    "font.family": "mononoki Nerd Font Mono",
    "font.serif": [],
    #  "font.cursive": ["mononoki Nerd Font", "mononoki Nerd Font Mono"],
})
plt.savefig(os.path.join("../output", "gershgorin.pgf"))

plt.show()
```

```
def get_companion_matrix(poly):
    """
    Calculate the companion matrix for a normalized polynomial
    """
    n =len(poly)
    cmat =np.zeros((n -1, n -1))
    cmat[:, n -2] =(-poly[1:])[::-1]
    cmat[np.arange(1, n -1), np.arange(0, n -2)] =1

    return cmat


def get_formated_poly(poly):
    """
    Return the polynomial such that the coefficient of the maximum power of x
    is always 1
    """
    return poly /poly[0]


poly =np.array([2, 3, 1, 4, 3, 6])

norm_poly =get_formated_poly(poly)
matrix =get_companion_matrix(norm_poly)

plot_gershgorin_discs(matrix)
```

۳-۳ محاسبه‌ی بزرگترین ریشه‌ی چندجمله‌ای از نظر اندازه

به این منظور، کافیست الگوریتم توصیف شده در فصل قبل را پیاده سازی کنیم:

```
def power_iteration(A, num_iterations=1000, tol=1e-6):
    """
    Power iteration method for finding the dominant eigenvalue and eigenvector.

    Parameters:
    - A: Square matrix for which eigenvalues are to be calculated.
    - num_iterations: Maximum number of iterations (default: 1000).
    - tol: Tolerance to determine convergence (default: 1e-6).

    Returns:
    - eigenvalue: Dominant eigenvalue.
    - eigenvector: Corresponding eigenvector.
    """
    n =A.shape[0]

    # Initialize a random vector
    v =np.random.rand(n)
    v =v /np.linalg.norm(v) # Normalize the vector

    for i in range(num_iterations):
        Av =np.dot(A, v)
        eigenvalue =np.dot(v, Av)
        v =Av /np.linalg.norm(Av)

        # Check for convergence
        if np.abs(np.dot(Av, v) -eigenvalue) <tol:
            break
```

```
    return eigenvalue, v
```

```
poly =np.array([2, 3, 1, 4, 3, 6])
norm_poly =get_formated_poly(poly)
matrix =get_companion_matrix(norm_poly)
root, _ =power_iteration(matrix)
```

برای تست نتیجه، می‌توانیم مقدار چندجمله‌ای در این نقطه‌را به دست آوریم:

```
def eval_poly(poly, x):
    cur_x =1
    total =0
    for a in poly[::-1]:
        total +=cur_x *a
        cur_x *=x
    return total
eval_poly(poly, root)
```

کتاب‌نامه

[1] Balancing a matrix for calculation of eigenvalues and eigenvectors. *Numerische Mathematik*, 13(4):293–304, 1969.

[2] Edelman, Alan and Murakami, Hiroshi. Polynomial roots from companion matrix eigenvalues. *Mathematics of Computation*, 64(210):763–776, 1995.

[3] Fujiwara, Eiji, Yashiro, Mitsuhiko, and Furuya, Tsuneo. 7 - applications to computer systems. In Imai, Hideki, editor, *Essentials of Error-Control Coding Techniques*, pages 171–268. Academic Press, 1990.

[4] Golub, Gene H and Van Loan, Charles F. *Matrix computations*. JHU press, 2013.

[5] James, Rodney, Langou, Julien, and Lowery, Bradley R. On matrix balancing and eigenvector computation. *arXiv preprint arXiv:1401.5766*, 2014.

[6] Moler, Cleve. Cleve's corner: Roots-of polynomials, that is. *The MathWorks Newsletter*, 5(1):8–9, 1991.

[7] Osborne, EE. On pre-conditioning of matrices. *Journal of the ACM (JACM)*, 7(4):338–345, 1960.

پیوست