UNIVERSITAT DE BARCELONA
Facultat de Matemàtiques
i Informàtica

GRAU DE MATEMÀTIQUES

Treball final de grau

TEOREMA DE STURM MATRICIAL I IMPLEMENTACIÓ EN PYTHON

Autor: Pau Sallent Jurado

Director: Dr. Carlos D'Andrea
Realitzat a: Departament de Matemàtiques
i Informàtica

Barcelona, 12 de juny de 2022

Abstract

Sturm's Theorem quantifies the real roots of a polinomial inside a given interval. We review the article by Kaiwen Hou and Bin Li which presents a demonstration of this theorem using matrix theory. We also implement the algorithm they describe in Python.

Resum

El Teorema de Sturm quantifica les arrels reals d'un polinomi dins un interval donat. En aquest treball revisem l'article de Kaiwen Hou i Bin Li que presenta una demostració del teorema mitjançant la teoria de matrius. També implementem l'algoritme que s'hi descriu en Python.

2020 Mathematics Subject Classification. 11B83, 15A15, 26C10, 90C23

Agraïments

Vull agrair al meu tutor, el Doctor Carlos D'Andrea, sense les seves indicacions inicials no hauria sigut possible aquest treball.

També als companys que han fet el tfg alhora que jo, per compartir les respostes als seus dubtes i per tots els consells.

Finalment a la meva psicòloga per ajudar-me, amb la organització i a mantenir la motivació per acabar el treball.

Índex

1 Introducció **1**

2 Teorema de Sturm **2**

2.1 Definicions bàsiques 2

3 Matriu de Sturm i demostració matricial **6**

3.1 Preliminars 6

3.2 Matriu de Sturm i Teorema de Sturm 8

3.3 Demostració del Teorema de Sturm 13

4 Implementació en Python **16**

4.1 Objectius 16

4.2 Per què Python: l'extensió NumPy 16

4.3 Funcionament del programa 17

5 Conclusions **23**

1 Introducció

El projecte

En matemàtiques, tot i que a primera vista podem trobar dos camps amb aparentment poca relació, sovint veiem que diferents elements de les matemàtiques tenen connexions més estretes del que sembla.

El 1835, el matemàtic francès Jacques Charles François Sturm va descobrir el teorema que porta el seu nom [2]. En aquest treball definirem els elements bàsics per poder enunciar i entendre el teorema de Sturm i revisarem i ampliarem la demostració que Kaiwen Hou i Bin Li donen al seu article [1].

Pel teorema de Sturm, el nombre d'arrels reals diferents d'un polinomi donat $f(x)$ en qualsevol interval $(a, b]$ es pot expressar pel nombre de variacions de signe de la cadena de Sturm als extrems de l'interval. Construint la "Matriu de Sturm", una matriu simètrica associada a $f(x) \in \mathbb{R}[x]$, les variacions del signe de $f(x)$ es poden caracteritzar per l'index d'inèrcia negatiu d'aquesta matriu.

També implementarem l'algoritme que ofereixen pel càlcul de la cadena de Sturm i matriu de Sturm d'un polinomi en un programa de Python que ens servirà per detectar el nombre d'arrels reals donat un interval.

Estructura de la Memòria

En aquest primer capítol donarem les definicions necessàries per establir una base matemàtica on estructurar el Teorema de Sturm i donarem també una demostració elemental del teorema.

Al llarg del segon capítol es fa una exposició dels conceptes que es desenvolupen a [1] i es proposen ampliacions de les demostracions de K. Hou i B. Li. També hi inclourem exemples d'aquests elements amb polinomis concrets.

Pel que fa al tercer capítol, comentem part per part la realització del programa de Python inclòs a l'Annex I, també amb explicacions de com implementar les funcions pròpies de Python i el funcionament del codi que hem creat.

2 Teorema de Sturm

Aquest capítol comprèn les definicions necessàries per enunciar i demostrar el Teorema de Sturm clàssic.

2.1 Definicions bàsiques

Proposició 2.1. *Siguin K un cos i $f(X), g(X)$ polinomis tals que $g(X) \neq 0$. Llavors, existeixen polinomis $q(X), r(X) \in K[X]$, únics, tals que*

$$f(X) = g(X)q(X) + r(X) \text{ i } gr(r(X)) < gr(g(X)).$$

On $gr(p(X))$ indica el grau del polinomi $p(X)$.

Demostració. [3] En el cas que $gr(f(X)) < gr(g(X))$, l'existència és gairebé immediata. En efecte, posem $q(X) := 0$, $r(X) := f(X)$; llavors, és clar que se satisfan les propietats enunciades: $f(X) = g(X)q(X) + r(X)$ i $gr(r(X)) < gr(g(X))$. Suposem, doncs, que $gr(f(X)) \leq gr(g(X))$.

Siguin $f(X) =: a_0 + a_1X + \cdots + a_nX^n$, $a_0, a_1, \ldots, a_n \in K$, $g(X) =: b_0 + b_1X + \cdots + b_mX^m$, $b_0, b_1, \ldots, b_m \in K$, amb $a_n \neq 0$, $b_m \neq 0$, i $n \geq m$. Llavors, el polinomi

$$f(X) - g(X)\tfrac{a_n}{b_m}X^{n-m} =: c_0 + c_1X + \cdots + c_{n-1}X^{n-1},$$

on $c_0, \ldots, c_{n-1} \in K$, és de grau estrictament menor que el grau de $f(X)$; repetint el mateix procés, el polinomi

$$f(X) - g(X)\tfrac{a_n}{b_m}X^{n-m} - g(X)\tfrac{c_{n-1}}{b_m}X^{n-m-1} = f(X) - g(X)(\tfrac{a_n}{b_m}X^{n-m} - \tfrac{c_{n-1}}{b_m}X^{n-m-1})$$

és un polinomi de grau menor o igual que $n-2$. Doncs, podem calcular recursivament un polinomi

$$q(X) := \tfrac{a_n}{b_m}X^{n-m} - \tfrac{c_{n-1}}{b_m}X^{n-m-1} + \ldots$$

El procés s'atura quan el grau de la diferència $f(X) - g(X)q(X)$ és menor que el grau, m, de $g(X)$. Definim el polinomi $r(X)$ com aquesta diferència: $r(X) := f(X) - g(X)q(X)$. Llavors, les propietats enunciades se satisfan per definició de $q(X)$ i de $r(X)$.

Demostrem, ara, la unicitat. Suposem que

$$\begin{aligned} f(X) &= g(X)q_1(X) + r_1(X),\ gr(r_1(X)) < gr(g(X)), \\ f(X) &= g(X)q_2(X) + r_2(X),\ gr(r_2(X)) < gr(g(X)), \end{aligned}$$

on $q_1(X), q_2(X), r_1(X), r_2(X) \in K[X]$. Es té que $r_2(X) - r_1(X) = g(X)(q1(X) - q2(X))$, de manera que el grau de $r_2(X) - r_1(X)$, que és menor estricte que el grau de $g(X)$, coincideix amb el grau del producte del polinomi $g(X)$ pel polinomi $q_1(X) - q_2(X)$; però, si $q_1(X) - q_2(X) \neq 0$, el grau d'aquest producte és la suma dels graus dels dos factors, de manera que és m és gran o igual que el grau de $g(X)$. Aquesta contradicció obliga que sigui $q_1(X) = q_1(X)$ i, en conseqüència, $r_1(X) = r_2(X)$, com volíem provar. □

Definició 2.2. *S'anomena la divisió entera del polinomi $f(X)$ per $g(X)$ al càlcul dels polinomis $q(X)$ i $r(X)$ que apareixen a la proposició anterior.*

L'algoritme d'Euclides per a polinomis fa servir l'existència de quocients i residus en la divisió entera entre polinomis i també la igualtat

$$mdc(f(X), g(X)) = mcd(f(X) - g(X)q(X), g(X)),$$

que és certa per qualssevol $f(X), g(X), q(X)$ polinomis de $K[X]$, per calcular el màxim comú divisor entre dos polinomis. L'algoritme calcula la seqüència següent:

$$\begin{gathered} f_0(X) = d_1(X)f_1(X) + f_2(X),\ gr(f_2) < gr(f_1), \\ f_1(X) = d_2(X)f_2(X) + f_3(X),\ gr(f_3) < gr(f_2), \\ \vdots \\ f_{m-2}(X) = d_{m-1}(X)f_{m-1}(X) + f_m(X),\ gr(f_m) < gr(f_{m-1}), \\ f_{m-1}(X) = d_m(X)f_m(X) \end{gathered}$$

i se satisfà:

$mcd(f_0(X), f_1(X)) = mcd(f_0(X) - f_1(X)d_1(X), f_1(X)) = mcd(f_2(X), f_1(X))$.
I també, per tot $1 \leq i \leq m-1$:

$$\begin{gathered} mcd(f_{i-1}(X), f_i(X)) = mcd(f_{i+1}(X), f_i(X)) = mcd(f_i(X), f_{i+1}(X)) \implies \\ \implies mcd(f_0(X), f_1(X)) = mcd(f_m(X), f_{m-1}(X)) = \\ = mcd(f_m(X), f_{m-1}(X) - f_m(X)d_m(X)) = mcd(f_m(X), 0)) = f_m(X) \end{gathered}$$

Definició 2.3. *Donat un polinomi no constant $f_0(X) \in K[X]$, anomenem la cadena de Sturm, a la seqüència $f_0(X), \ldots, f_m(X)$ que calculem mitjançant l'algoritme d'Euclides amb una petita modificació:*

$$\begin{gathered} f_1(X) := f_0'(X) \\ f_0(X) = d_1(X)f_1(X) - f_2(X),\ gr(f_2) < gr(f_1), \\ f_1(X) = d_2(X)f_2(X) - f_3(X),\ gr(f_3) < gr(f_2), \\ \vdots \\ f_{m-2}(X) = d_{m-1}(X)f_{m-1}(X) - f_m(X),\ gr(f_m) < gr(f_{m-1}), \\ f_{m-1}(X) = d_m(X)f_m(X) \end{gathered}$$

Nota: D'ara endavant utilitzarem x minúscula en comptes de X com a variable dels polinomis i les funcions.

Definició 2.4. *La funció delta de Kronecker i la funció signe, notades $\delta_{i,j}$ i sgn(x) respectivament, son funcions definides com:*

$$\delta_{i,j} = \begin{cases} 1 & si \quad i \neq j \\ 0 & si \quad i = j \end{cases}$$

$$sgn(x) = \begin{cases} 1 & si \quad x > 0 \\ 0 & si \quad x = 0 \\ -1 & si \quad x < 0 \end{cases}$$

Per una seqüència de nombres reals no nuls $c_1, c_2, ..., c_n$, denotem per $\sigma(c_1, ..., c_n)$ el nombre de variacions de signe de la seqüència, i.e.

$$\sigma(c_1, ..., c_n) = \textstyle\sum_{i=1}^{n-1} \delta_{-1, sgn(c_i c_{i+1})},$$

Per una seqüència de nombres reals $c_1, ..., c_n$ sigui $c_{i_1}, ..., c_{i_m}$ la subseqüència d'aquesta obtinguda eliminant tots els zeros i definim

$$\sigma(c_1, ..., c_n) = \sigma(c_{i_1}, ..., c_{i_m}).$$

Sigui $V_f(x) = \sigma(f_0(x), f_1(x), ..., f_m(x))$.

Teorema 2.5. *(Teorema de Sturm): Per un polinomi no constant $f(x) \in \mathbb{R}[x]$ i dos nombres reals $a < b$, el nombre d'arrels reals diferents a l'interval $(a, b]$ és $V_f(a) - V_f(b)$ si ni a ni b son arrels múltiples de $f(x)$. A continuació es dona una demostració clàssica del Teorema de Sturm [4].*

Demostració. : Estudiarem com varia la funció $V_f(x)$ a mesura que x avança en l'interval (a, b). Primer fem un pas previ: Afirmem que podem suposar sense pèrdua de generalitat que $f(x)$ té totes les seves arrels simples. En efecte si això no fos així, aleshores $f(x)$ i $f'(x)$ tindrien un factor en comú $p(x)$. Aquest factor també seria múltiple de $f_2(x)$ perquè

$$f_2(x) = c(x)f'(x) - f(x), \text{ amb } c(x) \in \mathbb{R}[x],$$

i com que

$$f_3(x) = c^*(x)f_2(x) - f'(x), \text{ amb } c^*(x) \in \mathbb{R}[x],$$

inductivament es té que $p(x)$ és un factor de $f_i(x)$ per tot $i = 0, 1, ...$ D'altra banda, és fàcil veure que

$$V(p(x)A_0(x), p(x)A_1(x), p(x)A_2(x), ...) = V(A_0(x), A_1(x), A_2(x), ...)$$

si $p(x) \neq 0$. Doncs com que V només estudia les variacions de signe tenim:

$$V(p(x)A_0(x), p(x)A_1(x), ...) = V(-A_0(x), -A_1(x), ...) = V(A_0(x), A_1(x), ...)$$

si $p(x) < 0$ i

$$V(p(x)A_0(x), p(x)A_1(x), ...) = V(A_0(x), A_1(x), ...)$$

si $p(x) > 0$. Això permet treballar amb polinomis $f(x)$ que no tinguin cap factor en comú amb $f'(x)$. Notar que això també implica que totes les arrels de $f(x)$ seran simples. Això és perquè si $f(x)$ té una arrel k amb multiplicitat $\alpha > 1$ tenim $f(x) = (x - k)p(x)$ per algun $p(x) \in \mathbb{R}[x]$ tal que $p(k) = 0$. Aleshores $f'(x) = p(x) + (x - k)p'(x)$ i és evident que $f'(k) = 0$, és a dir que f i f' tindrien el factor $(x - k)$ comú.

Ara observem que el fet que f i f' no tinguin arrels comunes implica que per tot $i = 0, 1, ...$ no hi ha arrels comunes entre f_i i f_{i+1}. En efecte si α és una arrel de f_i i f_{i+1}, també ho serà de f_{i-1} perquè $f_{i-1} = c^* f_i - f_{i+1}$, amb $c^* \in \mathbb{R}[x]$, i aleshores inductivament s'arriba a $f_0(\alpha) = f_1(\alpha) = 0$, una contradicció.

Fetes aquestes observacions, comencem a estudiar la variació de signes de la successió $(f_0(x), f_1(x), ...)$. - Si existeix $\alpha \in (x_0, x_1)$ tal que $f(\alpha) = 0$, aleshores com que $f'(\alpha)$ serà diferent de zero, f creixerà o decreixerà en un entorn d'α. En tot cas, f_0 canvia de signe al passar per α, però no passa el mateix amb f_1. Aquí hi ha una variació de signe. - Suposem que a més existeix algun $i > 2$ tal que $f_i(\alpha) = 0$. Com que $f_{i-1}(x) = c^*(x)f_i(x) - f_{i+1}(x)$, aleshores es té

$$f_{i-1}(\alpha) = -f_{i+1}(\alpha),$$

és a dir, els valors de f_{i-1} i f_{i+1} en un entorn d'α són de signes oposats (i a més sabem que cap d'ells pot ser igual a zero). Això diu que, independentment del valor que tenia f_i en un entorn d'α, el nombre de variacions de signe en la terna f_{i-1}, f_i, f_{i+1} no canvia en atravessar α. En conclusió, l'únic canvi en la successió de signes de $(f_0(x), f_1(x), ...)$ es produeix quan s'atravessa una arrel de f i això és el que volíem demostrar. □

Observació 2.6. És interessant notar que el nombre de passos per calcular la cadena de Sturm d'un polinomi de grau n és finit, doncs a la demostració de l'existència del quocient i el residu de la divisió entera de polinomis es dona un algoritme per calcular-los amb un nombre finit de passos, i hem de fer un nombre finit de divisions enteres per trobar la cadena de Sturm.

D'ara endavant introduïm els conceptes que es desenvolupen a [1].

3 Matriu de Sturm i demostració matricial

3.1 Preliminars

Donada una matriu real simètrica A, la forma quadràtica definida per A es pot convertir a una matriu diagonal mitjançant una transformació de coordenades no singular. La llei d'inèrcia de Sylvester garanteix que el nombre de coeficients positius (negatius) a la forma diagonal es un invariant anomenat l'index positiu (negatiu) d'inèrcia denotat per $p(A)$ (respectivament $q(A)$). Val la pena notar que $p(A)$ i $q(A)$ és el nombre de valors propis positius i negatius (comptats amb multiplicitats) d'A respectivament.

Lema 3.1. *Si A es una matriu real simètrica $n\times n$ de rang $r(A) \geq n-1$ i $B = \begin{bmatrix} A & \alpha \\ \alpha^T & b \end{bmatrix}$*

amb $\alpha \in \mathbb{R}^n$ i $b \in \mathbb{R}$ tals que els determinants $|A|$ i $|B|$ satisfan $|A|^2 + |B|^2 \neq 0$, aleshores

$$q(B) = \begin{cases} q(A) & si \quad |A|\,|B| > 0 \vee |B| = 0 \\ \\ q(A)+1 & si \quad |A|\,|B| < 0 \vee |A| = 0 \end{cases}$$

Demostració. Provem-ho pel cas $|A|\,|B| \neq 0$. Tenim que $q(A) = q(MAM^T)$ (respectivament per $p(A)$), per qualsevol M invertible. A més, pel procés d'ortogonalització existeix una matriu M_0 tal que:

$$M_0 A M_0^T = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_m \end{bmatrix}$$

On $\lambda_1, ..., \lambda_m$ són els valors propis de la matriu A. Ara, se satisfà:

$$\begin{bmatrix} M_0 & \\ & 1 \end{bmatrix} B \begin{bmatrix} M_0^T & \\ & 1 \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & & b_1 \\ & \ddots & & & b_2 \\ & & \ddots & & \vdots \\ & & & \lambda_n & b_n \\ b_1 & b_2 & \dots & b_n & b \end{bmatrix}$$

A la última fila li restem la fila i multiplicada per $\frac{b_i}{\lambda_i}$, $\forall i$, $1 \leq i \leq n$. Ens quedarem amb una matriu com la següent:

$$\begin{bmatrix} \lambda_1 & & & & b_1 \\ & \ddots & & & b_2 \\ & & \ddots & & \vdots \\ & & & \lambda_n & b_n \\ 0 & 0 & \dots & 0 & c \end{bmatrix}$$

amb $c \neq 0$ Com que és una matriu triangular superior, els valors propis d'aquesta matriu són $\lambda_1, \lambda_2, \dots, \lambda_n, c$ i com que aquesta matriu conjuga amb B, tenim que si $c < 0$, aleshores

$|B|\,|A| < 0$ i a més, $q(B) = q(A)+1$. Si, per altra banda, $c > 0$, aleshores tenim $|B|\,|A| > 0$ i també $q(B) = q(A)$.

Ara si $|B| = 0$, la condició $|A|^2 + |B|^2 \neq 0$ implica que A és invertible. Tenim

$$\begin{bmatrix} I & 0 \\ -\alpha^T A^{-1} & 1 \end{bmatrix} \begin{bmatrix} A & \alpha \\ \alpha^T & b \end{bmatrix} \begin{bmatrix} I & -A^{-1}\alpha \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} A & 0 \\ 0 & b - \alpha^T A^{-1}\alpha \end{bmatrix}$$

Prenent determinant als dos costats tenim que $b - \alpha^T A^{-1}\alpha = 0$. Per tant, B es conjuga amb
$\begin{bmatrix} A & 0 \\ 0 & 0 \end{bmatrix}$
que implica $q(B) = q(A)$. Ara si $|A| = 0$, aleshores $rang(A) = n - 1$ i $|B| \neq 0$. Existeix una matriu ortogonal Q tal que

$$Q^T AQ = \begin{bmatrix} \lambda_1 & & & & \\ & \lambda_2 & & & \\ & & \ddots & & \\ & & & \lambda_{n-1} & \\ & & & & 0 \end{bmatrix}$$

on $\lambda_i (1 \leq i \leq n-1)$ són els valors propis diferents de 0 d'A. Aleshores tenim

$$\begin{bmatrix} Q^T & \\ & 1 \end{bmatrix} B \begin{bmatrix} Q & \\ & 1 \end{bmatrix} = \begin{bmatrix} \lambda_1 & & & & b_1 \\ & \ddots & & & b_2 \\ & & \lambda_{n-1} & & \vdots \\ & & & 0 & b_n \\ b_1 & b_2 & \dots & b_n & b \end{bmatrix}$$

que conjuga a una matriu de la forma

$$\begin{bmatrix} \lambda_1 & & & & \\ & \ddots & & & \\ & & \lambda_{n-1} & & \vdots \\ & & & 0 & c \\ & & \dots & c & d \end{bmatrix}.$$

Com que $|B| \neq 0$, $c \neq 0$ i per tant els dos valors propis μ_1, μ_2 de $\begin{bmatrix} 0 & c \\ c & d \end{bmatrix}$ satisfan

$$\mu_1 \mu_2 = -c^2 < 0$$

Això implica que $q(B) = q(A) + 1$.

□

Donada una matriu A $n \times n$ i $I \subseteq 1, ..., n$, sigui A_I la submatriu principal d'A les columnes i files de la qual estan indexades per I. El determinant d'A_I es diu el menor principal d'A d'ordre $\#I$, on $\#I$ denota la cardinalitat d'I.

Definició 3.2. *Sigui A una matriu $n \times n$ sobre $\mathbb{R}$.*

(i) diem que $D_1, D_2, ..., D_n$ és la seqüència principal de menors d'A, si $D_i = |A_{I_i}|$ on $I_i \subset I_{i+1} \subseteq \{1, ..., n\}$ i $\#I_i = i$.

(ii) Una seqüència principal de menors $D_1, D_2, ..., D_n$ es diu que és normal si $D_r \neq 0$ amb $r = r(A)$ i per $1 \leq i \leq r-1$, dos menors consecutius qualssevol D_i, D_{i+1} no són els dos 0.

Aplicant el Lema 3.1 obtenim el següent lema.

Lema 3.3. *Donada una matriu $n \times n$ simètrica A de rang $r(A) = r$, si $D_1, D_2, ..., D_n$ és una seqüència principal de menors normal d'A, aleshores*

$$q(A) = \delta_{-1,sgn(D_1)} + \sum_{i=1}^{r-1} \delta_{-1,sgn(D_i D_{i+1})} + \sum_{i=1}^{r-1} \delta_{0,D_i}.$$

Demostració. Veiem que $q(A_{I_1}) = \delta_{-1,D_1}$, és a dir que la fórmula funciona per $r = 1$. Ara suposem que funciona per $r = m$ i veurem que també funciona per $r = m + 1$. Com que A_{I_i} és una matriu simètrica per tot i i com que D_i és una seqüència de menors principal i per tant $D_i^2 + D_{i+1}^2 \neq 0$ i a més $r(A_{i+1}) \geq i$, aplicant el lema 2.3.1 se satisfà:

$$q(A_{I_{m+1}}) = q(A_{I_m}) + \delta_{-1,sgn(D_m D_{m+1})} + \delta_{0,D_m}$$

És a dir,

$$q(A_{I_{m+1}}) = \delta_{-1,sgn(D_1)} + \sum_{i=1}^{m} \delta_{-1,sgn(D_i D_{i+1})} + \sum_{i=1}^{m} \delta_{0,D_i}$$

Per tant se satisfà:

$$q(A_{I_r}) = \delta_{-1,sgn(D_1)} + \sum_{i=1}^{r-1} \delta_{-1,sgn(D_i D_{i+1})} + \sum_{i=1}^{r-1} \delta_{0,D_i}$$

Com que $r(A) = r$ sabem que per tot $i > r$, $D_i = 0$, per tant també aplicant el lema, $q(A_{I_i}) = q(A_{I_{i+1}})$, és a dir:

$$q(A) = q(A_{I_n}) = q(A_{I_r}) = \delta_{-1,sgn(D_1)} + \sum_{i=1}^{r-1} \delta_{-1,sgn(D_i D_{i+1})} + \sum_{i=1}^{r-1} \delta_{0,D_i}$$

□

Observació 3.4. Val la pena senyalar que es pot demostrar, per qualsevol matriu simètrica real A, existeix una seqüència principal de menors per la que es pot determinar el seu índex d'inèrcia positiu i el negatiu.

3.2 Matriu de Sturm i Teorema de Sturm

Sigui $f_0(x) = f(x)$ i $f_1(x) = g(x)$. La cadena de Sturm $f_0, ..., f_m$ associada a (f, g) s'obté de la manera següent:

$$
\begin{gathered}
f_0 = d_1(x)f_1(x) - f_2(x), deg(f_2) < deg(f_1), \\
f_1 = d_2(x)f_2(x) - f_3(x), deg(f_3) < deg(f_2), \\
\vdots \\
f_{m-2} = d_{m-1}(x)f_{m-1}(x) - f_m(x), deg(f_m) < deg(f_{m-1}), \\
f_{m-1} = d_m(x)f_m(x).
\end{gathered}
$$

Les següents afirmacions sobre la cadena de Sturm son evidents.
(a) Si a és una arrel comuna de $f(x)$ i $g(x)$, aleshores $f_i(a) = 0$ per tot $1 \leq i \leq m$; altrament, qualssevol polinomis consecutius f_i i f_{i+1} de la cadena no tenen arrels comunes.
(b) $f_m(x) = mcd(f(x), g(x))$, doncs a és una arrel comuna de $f(x)$ i $g(x)$ si, i només si $f_m(a) = 0$.
(c) Si $f_i(a) = 0$ per algun $1 \leq i \leq m-1$ i $f_m(a) \neq 0$, aleshores, $f_{i-1}(a)f_{i+1}(a) < 0$.

Definició 3.5. *Per $f(x), g(x) \in \mathbb{R}[x]$, la matriu de Sturm associada a $(f(x), g(x))$ es defineix com:*

$$
\begin{bmatrix}
d_1(x) & 1 & & \\
1 & d_2(x) & \ddots & \\
 & \ddots & \ddots & 1 \\
 & & 1 & d_m(x)
\end{bmatrix},
$$

i s'escriu com $S_{f,g}(x)$.

Exemple 3.6. Sigui $f(x) \in \mathbb{R}[x]$, $f_0(x) = f(x) = x^3 - 7x + 1$, calculem la cadena de Sturm associada a $(f(x), f'(x))$:

$$
\begin{gathered}
f_1(x) := f'(x) = 3x^2 - 7 \\
f_0(x) = d_1(x) * f_1(x) - f_2(x) \\
x^3(x) = d_1(x) * (3x^2 - 7) - f_2(x),
\end{gathered}
$$

i calculant la divisió entera entre $x^3 - 7x + 1$ i $3x^2 - 7$ tenim

$$
\begin{gathered}
d_1(x) = \tfrac{1}{3}x, \\
f_2(x) = \tfrac{14}{3}x - 1
\end{gathered}
$$

i continuem

$$
\begin{gathered}
f_1(x) = d_2(x) * f_2(x) - f_3(x) \\
3x^2 - 7 = d_2(x) * \tfrac{14}{3}x - 1 - f_3(x),
\end{gathered}
$$

fem el mateix

$$
\begin{gathered}
d_2(x) = \tfrac{-9}{14}x - \tfrac{27}{196} \\
f_3(x) = \tfrac{1345}{196} \\
f_2(x) = d_3(x) * f_3(x) - f_4(x) \\
\tfrac{14}{3}x - 1 = d_3(x) * \tfrac{1345}{196} \\
d_3(x) = \tfrac{2744}{4035}x - \tfrac{196}{1345}
\end{gathered}
$$

En definitiva, tenim que la cadena de Sturm de $f(x) = x^3 - 7x + 1$ és:

$$\{x^3 - 7x + 1,\ 3x^2 - 7,\ \tfrac{14}{3}x - 1,\ \tfrac{1345}{196}\}$$

A més, com que coneixem $d_1(x)$, $d_2(x)$, $d_3(x)$ podem trobar la matriu de Sturm:

$$S_{f,f'}(x) = \begin{bmatrix} \frac{1}{3}x & 1 & 0 \\ 1 & \frac{-9}{14}x - \frac{27}{196} & 1 \\ 0 & 1 & \frac{2744}{4035}x - \frac{196}{1345} \end{bmatrix}.$$

Notem que $r(S_{f,g}(x)) \geq m - 1$ i per qualsevol divisor comú $d(x)$ de $f(x)$ i $g(x)$,

$$S_{f,g}(x) = S_{\frac{f}{d},\frac{g}{d}}(x)$$

És evident, perquè si per $1 \leq i \leq m$ definim:

$$s_i(x) := \begin{vmatrix} d_1(x) & 1 & & \\ 1 & d_2(x) & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & d_i(x) \end{vmatrix},$$

per una propietat de les matrius tridiagonals [5], se satisfà en aquest cas $s_i(x) = d_i(x)s_{i-1}(x) - s_{i-2}(x)$, amb $s_0(x) := 1$, $s_{-1}(x) := 0$. Aleshores no podria ser $s_m(x) = s_{m-1}(x) = 0$ perquè tindríem $s_{m-2}(x) = d_m s_{m-1}(x) - s_m(x) = 0$ i successivament $s_i(x) = 0$ amb $1 \leq i \leq m$ però sabem que $s_1(x) = d_1(x) \neq 0$, per tant, $r(S_{f,g}(x)) \geq m - 1$.

Nota: En la referència [5], trobem la següent fórmula pel càlcul del determinant de matrius tridiagonals de la forma:

$$\begin{vmatrix} a_1 & c_1 & & \\ b_1 & a_2 & \ddots & \\ & \ddots & \ddots & c_{n-1} \\ & & b_{n-1} & a_n \end{vmatrix} =: D_n,$$

$$D_n = a_n D_{n-1} - c_{n-1}b_{n-1}D_{n-2}$$

D'altra banda, si calculem la cadena de Sturm associada a $(\frac{f}{d}, \frac{g}{d})$ amb $d(x)$ un divisor comú de $f(x)$ i $g(x)$ (i per tant de f_i per tot i) obtenim:

$$\begin{gathered} \tfrac{f_0(x)}{d(x)} = d_1(x)\tfrac{f_1(x)}{d(x)} - \tfrac{f_2(x)}{d(x)},\ deg(\tfrac{f_2}{d}) < deg(\tfrac{f_1}{d}) \leftrightarrow \\ \leftrightarrow f_0 = d_1(x)f_1(x) - f_2(x),\ deg(f_2) < deg(f_1) \end{gathered}$$

i així successivament:

$$\begin{gathered} \tfrac{f_{i-1}(x)}{d(x)} = d_i(x)\tfrac{f_i(x)}{d(x)} - \tfrac{f_{i+1}(x)}{d(x)},\ deg(\tfrac{f_{i+1}}{d}) < deg(\tfrac{f_i}{d}) \leftrightarrow \\ \leftrightarrow f_{i-1} = d_i(x)f_i(x) - f_{i+1}(x),\ deg(f_{i+1}) < deg(f_i) \end{gathered}$$

Definició 3.7. *Definim la cadena de Sturm refinada associada a* $(f(x), g(x))$ *com*

$$\tilde{f_0}, \tilde{f_1}, ..., \tilde{f_m},$$

on $\tilde{f_i} = \frac{f_i}{f_m}$.

Se satisfà $S_{f_0,f_1}(x) = S_{\tilde{f_0},\tilde{f_1}}(x)$. Sigui $V_{f,g}(x)$ el nombre de variacions de signe de $f_0(x), f_1(x), ..., f_m(x)$, i.e.

$$V_{f,g}(x) = \sigma(f_0(x), f_1(x), ..., f_m(x)).$$

Veiem que

$$V_{f,g}(a) = V_{\tilde{f},\tilde{g}}(a)$$

amb $\tilde{g} = \tilde{f_1}$ si a no és una arrel comuna de $f(x)$ i $g(x)$.

Exemple 3.8. Calculem la cadena de Sturm refinada pel nostre exemple anterior. Tenim que per $0 \leq i \leq 2$

$$\begin{gathered}\tilde{f_i}(x) = \frac{f_i(x)}{f_3(x)} \text{ i } \tilde{f_3}(x) = 1 \\ \tilde{f_0}(x) = \frac{196}{1345}x^3 - \frac{1372}{1345}x + \frac{196}{1345} \\ \tilde{f_1}(x) = \frac{588}{1345}x^2 - \frac{1372}{1345} \\ \tilde{f_2}(x) = \frac{2744}{4035}x - \frac{196}{1345}\end{gathered}$$

i en efecte, se satisfà

$$\begin{gathered}\tilde{f_0}(x) = d_1(x) * \tilde{f_1}(x) - \tilde{f_2}(x) \\ \tilde{f_1}(x) = d_2(x) * \tilde{f_2}(x) - \tilde{f_3}(x) \\ \tilde{f_2}(x) = d_3(x) * \tilde{f_3}(x) = d_3(x),\end{gathered}$$

doncs si substituïm:

$$\begin{gathered}\frac{196}{1345}x^3 - \frac{1372}{1345}x + \frac{196}{1345} = \frac{1}{3}x * (\frac{588}{1345}x^2 - \frac{1372}{1345}) - (\frac{2744}{4035}x - \frac{196}{1345}) \\ \frac{588}{1345}x^2 - \frac{1372}{1345} = (\frac{-9}{14}x - \frac{27}{196}) * \frac{2744}{4035}x - \frac{196}{1345} - 1 \\ \frac{2744}{4035}x - \frac{196}{1345} = (\frac{2744}{4035}x - \frac{196}{1345}) * 1\end{gathered}$$

Simplificar $\frac{1}{3} \cdot x \cdot \left(\frac{588}{1345}x^2 - \frac{1372}{1345}\right) - \left(\frac{2744}{4035}x - \frac{196}{1345}\right)$: $\frac{196x^3}{1345} - \frac{1372x}{1345} + \frac{196}{1345}$

Simplificar $\left(-\frac{9}{14}x - \frac{27}{196}\right) \times \left(-\frac{2744}{4035}x + \frac{196}{1345}\right) - 1$: $\frac{588x^2}{1345} - \frac{1372}{1345}$

Figura 1: Captures dels càlculs de les igualtats anteriors [8]

Considerem el següent menor principal de $S_{f,g}(x)$,

$$D_i = \left|S_{f,g}(x)_{\{m-i+1,m-i+2,...,m\}}\right|,$$

que és d'ordre i per $1 \leq i \leq m$.

Proposició 3.9. *Per $1 \leq i \leq m$, $D_i(x) = \tilde{f}_{m-i}(x)$.*

Demostració. Fem servir inducció en i. Veiem que $D_1(x) = d_m(x) = \frac{f_{m-1}(x)}{f_m(x)} = \tilde{f}_{m-1}(x)$ i

$$D_2(x) = \begin{vmatrix} d_{m-1}(x) & 1 \\ 1 & d_m(x) \end{vmatrix} = d_{m-1}(x)d_m(x) - 1 = \frac{f_{m-1}(x)d_{m-1}(x)}{f_m(x)} - 1 = \frac{f_{m-2}(x)+f_m(x)}{f_m(x)} - 1 = \frac{f_{m-2}(x)}{f_m(x)} = \tilde{f}_{m-2}(x)$$

Ara assumim que $D_k(x) = \tilde{f}_{m-k}(x)$ i que $D_{k+1}(x) = \tilde{f}_{m-k-1}(x)$.

$$D_{k+2}(x) = \begin{vmatrix} d_{m-k-1}(x) & 1 & & \\ 1 & d_{m-k}(x) & \ddots & \\ & \ddots & \ddots & 1 \\ & & 1 & d_m(x) \end{vmatrix}$$

Expandint el determinant per la primera fila, tenim:

$$D_{k+2}(x) = d_{m-k-1}(x)D_{k+1}(x) - \begin{vmatrix} 1 & 1 & & & \\ 0 & d_{m-k+1}(x) & 1 & & \\ & 1 & d_{m-k+2}(x) & \ddots & \\ \vdots & & \ddots & \ddots & 1 \\ 0 & & & 1 & d_m(x) \end{vmatrix}$$

i expandint ara per la primera columna ens queda:

$$D_{k+2}(x) = d_{m-k-1}(x)D_{k+1}(x) - D_k = d_{m-k-1}(x)\tilde{f}_{m-k-1}(x) - \tilde{f}_{m-k}(x) = \frac{d_{m-k-1}(x)f_{m-k-1}(x)-f_{m-k}(x)}{f_m(x)} = \frac{f_{m-k-2}(x)}{f_m(x)} = \tilde{f}_{m-k-2}(x)$$

□

Teorema 3.10. *$q(S_{f,g}(a)) = V_{f,g}(a)$ si a no és una arrel comuna de $f(x)$ i $g(x)$.*

Demostració. Per les afirmacions (a) i (b) sobre la cadena de Sturm (refinada), $D_1(a), \ldots, D_m(a)$ és una seqüència de menors principal de $S_{f,g}(a)$ si a no és una arrel comuna de $f(x)$ i $g(x)$. Per tant, pel Lema 3.3, tenim

$$q(S_{f,g}(a)) = \delta_{-1,sgn(D_1(a))} + \sum_{i=1}^{r-1} \delta_{-1,sgn(D_i(a)D_{i+1}(a))} + \sum_{i=1}^{r-1} \delta_{0,D_i(a)}. \tag{3.1}$$

amb $r = r(S_{f,g}(a))$. Notem que $r(S_{f,g}(a)) = m$ si a no és una arrel de $f(x)$, altrament, $r(S_{f,g}(a)) = m - 1$. En els dos casos (1) es pot reescriure com

$$q(S_{f,g}(a)) = \delta_{-1,sgn(D_1(a))} + \sum_{i=1}^{m-1} \delta_{-1,sgn(D_i(a)D_{i+1}(a))} + \sum_{i=1}^{m-1} \delta_{0,D_i(a)} \tag{3.2}$$

És a dir,

$$q(S_{f,g}(a)) = \delta_{-1,sgn(\tilde{f}_{m-1}(a))} + \sum_{i=1}^{m-1} \delta_{-1,sgn(\tilde{f}_{m-i}(a)\tilde{f}_{m-i-1}(a))} + \sum_{i=1}^{m-1} \delta_{0,\tilde{f}_{m-i}(a)}$$

$$= \sigma(1, \tilde{f}_{m-1}(a), \tilde{f}_{m-2}(a), \ldots, \tilde{f}_0(a))$$

$$= \sigma(\tilde{f}_0(a), \ldots, \tilde{f}_m(a))$$

$$= V_{f,g}(a),$$

El pas de la primera igualtat a la segona és degut a la propietat (c) de les cadenes de Sturm: Els canvis de signe en dos polinomis $\tilde{f}_i$ consecutius queden comptabilitzats a $\sum_{i=1}^{m-1} \delta_{-1,sgn(\tilde{f}_{m-i}(a)\tilde{f}_{m-i-1}(a))}$ i com que si $\tilde{f}_i(a) = 0$, aleshores, $\tilde{f}_{i-1}(a)\tilde{f}_{i+1}(a) < 0$ aquests canvis de signe queden comptabilitzats a $\sum_{i=1}^{m-1} \delta_{0,\tilde{f}_{m-i}(a)}$. Finalment, $\delta_{-1,sgn(\tilde{f}_{m-1}(a))} =$ $= \sigma(1, \tilde{f}_{m-1}(a))$.
□

3.3 Demostració del Teorema de Sturm

Sigui $a_1, \ldots, a_k \in \mathbb{R}$ totes les arrels de $\tilde{f}_0(x)$. Suposem que

$$a_1 < a_2 < \cdots < a_k.$$

Sigui I_i l'interval obert (a_i, a_{i+1}) per $0 \leq i \leq k$, amb la convenció $a_0 = -\infty$ i $a_{k+1} = +\infty$.

Proposició 3.11. *$q(S_{f,g}(x))$ és constant en cada interval I_i.*

Demostració. Com que $S_{f,g}(x)$ és una matriu simètrica real, per cada $x \in \mathbb{R}$ donada, els valors propis de $S_{f,g}(x)$

$$\lambda_1(x), \ldots, \lambda_m(x)$$

són nombres reals. Suposem $\lambda_1(x) \leq \cdots \leq \lambda_m(x)$. Tenim

$$|S_{f,g}(x)| = D_m(x) = \tilde{f}_0(x) = \lambda_1(x) \ldots \lambda_m(x) \neq 0$$

i el seu signe no canvia en l'interval I_i per la continuitat de $\tilde{f}_0(x)$. Per tant $\lambda_i(x)$ és diferent de 0 i no canvia de signe en I_i ja que és contínua en x. En conseqüència, el nombre de valors propis negatius de $S_{f,g}(x)$ és constant en I_i. □

Proposició 3.12. *Per $c \in \mathbb{R}$ tal que $\tilde{f}_0(c) = 0$, existeix $\epsilon > 0$ tal que $f(x)$ i $g(x)$ són diferents de 0 en $(c - \epsilon, c) \cup (c, c + \epsilon)$. A més*

$$q(S_{f,g}(x)) = \begin{cases} q(S_{f,g}(c)) & si \quad f(x)g(x) > 0 \\ q(S_{f,g}(c)) + 1 & si \quad f(x)g(x) < 0 \end{cases} \tag{3.3}$$

per $x \in (c - \epsilon, c) \cup (c, c + \epsilon)$.

Demostració. La primera part és evident per la continuitat de $f(x)$ i $g(x)$. Notem que $\tilde{f}_0(x) \neq 0$ amb $x \in (c - \epsilon, c) \cup (c, c + \epsilon)$ i també $\tilde{f}_1(x) \neq 0$ amb $x \in (c - \epsilon, c + \epsilon)$, doncs $\tilde{f}_1(c) \neq 0$.
Per simplicitat, denotem $T_{f,g}(x)$ la submatriu principal $S_{f,g}(x)_{\{2,3,\ldots,m\}}$ de $S_{f,g}(x)$. Per la Proposició 3.7,

$$|T_{f,g}(x)| = D_{m-1}(x) = \tilde{f}_1(x).$$

Com que el signe de $\tilde{f}_1(x)$ no canvia en $(c-\epsilon, c+\epsilon)$, és fàcil veure amb un argument semblant al de la Proposició 3.9 que $q(T_{f,g}(x))$ és constant en $(c-\epsilon, c+\epsilon)$. Pel Lema 3.1 se satisfà per $x \in (c-\epsilon, c) \cup (c, c+\epsilon)$,

$$q(S_{f,g}(x)) = \begin{cases} q(T_{f,g}(x)) & si \quad \tilde{f}_0(x)\tilde{f}_1(x) > 0 \\ q(T_{f,g}(x)) + 1 & si \quad \tilde{f}_0(x)\tilde{f}_1(x) < 0 \end{cases}$$

i també pel mateix Lema,

$$q(S_{f,g}(c)) = q(T_{f,g}(c)) = q(T_{f,g}(x))$$

Per tant, (3.3) és un resultat immediat ja que $f(x)g(x) > 0$ és equivalent a $\tilde{f}_0(x)\tilde{f}_1(x) > 0$ per $x \in (c-\epsilon, c) \cup (c, c+\epsilon)$. □

Sigui $q(S_{f,g}(x)) = q_i$ per $x \in I_i$ i $q(S_{f,g}(a_j)) = t_j$ amb $0 \leq i \leq k$ i $1 \leq j \leq k$. El següent corol·lari segueix de la proposició anterior.

Corołari 3.13. *Per* $1 \leq i \leq k$,

$$t_i = \begin{cases} q_{i-1} & si \quad f(x)g(x) > 0 \text{ en algun interval } (a_i - \epsilon, a_i) \\ q_{i-1} - 1 & si \quad f(x)g(x) < 0 \text{ en algun interval } (a_i - \epsilon, a_i) \end{cases}$$

$$= \begin{cases} q_i & si \quad f(x)g(x) > 0 \text{ en algun interval } (a_i, a_i + \epsilon) \\ q_i - 1 & si \quad f(x)g(x) < 0 \text{ en algun interval } (a_i, a_i + \epsilon) \end{cases}$$

Demostració. Només cal aplicar per casos la proposició anterior:

Sigui a_i una arrel de $f(x)$ i ϵ tal que $f(x)g(x)$ no canvia de signe quan $x \in (a_i - \epsilon, a_i)$. Si $f(x)g(x) < 0$, aleshores per la proposició tenim $q(S_{f,g}(x)) = q(S_{f,g}(a_i)) + 1$, és a dir, $q_{i-1} = t_i + 1$, per tant tenim $t_i = q_{i-1} - 1$. Si per contra, $f(x)g(x) > 0$, aleshores per la proposició, $q(S_{f,g}(x)) = q(S_{f,g}(a_i))$, o altrament, $t_i = q_{i-1}$.

D'altra banda i de manera semblant, sigui μ tal que $f(x)g(x)$ no canvia de signe quan $x \in (a_i, a_i + \mu)$. Si $f(x)g(x) < 0$, aleshores per la proposició tenim $q(S_{f,g}(x)) = q(S_{f,g}(a_i)) + 1$, és a dir, $q_i = t_i + 1$ i el que és el mateix, $t_i = q_i - 1$. Per últim, en el cas que $f(x)g(x) > 0$, aleshores per la proposició, $q(S_{f,g}(x)) = q(S_{f,g}(a_i))$; osigui $t_i = q_i$ □

Si $g(x)$ és la derivada de $f(x)$, $g(x) = f'(x)$, $\tilde{f}_0(x)$ té exactament les mateixes arrels que $f(x)$ amb multiplicitat 1. Com a funció polinòmica real, $f^2(x)$ arriba al seu mínim a a_i. És senzill veure que $(f^2(x))' = 2f(x)f'(x) > 0$ en algun interval $(a_i, a_i + \epsilon)$ i $2f(x)f'(x) < 0$ en algun interval $(a_i - \eta, a_i)$. Per tant, podem provar el Teorema de Sturm pel Corol·lari 3.11.

Teorema 3.14. *(Teorema de Sturm). Sigui* $a_1 < a_2 < \cdots < a_k$ *les arrels reals de* $f(x) \in \mathbb{R}[x]$, *i sigui* $g(x)$ *la derivada de* $f(x)$.

(i) $q(S_{f,g}(x))$ *és constant en cada interval* $(a_{i-1}, a_i]$ *amb valor* q_i *amb* $i = 1, 2, \ldots, k$, *on* $a_0 = -\infty$

(ii) $q_i - q_{i+1} = 1$ *per* $0 \leq i \leq k-1$.

(iii) Per qualsevol $a < b$, *el nombre d'arrels reals diferents de* $f(x)$ *a l'interval* $(a, b]$ *és* $q(S_{f,g}(a)) - q(S_{f,g}(b))$.

Demostració. La primera afirmació del teorema és evident, doncs és un cas particular de la Proposició 3.11.

Pel que fa a la segona, podem observar que en ser $g(x)$ la derivada de $f(x)$, per l'explicació anterior a l'enunciat del teorema se satisfà $(f^2(x))' = 2f(x)f'(x) > 0$ en algun interval $(a_i, a_i + \epsilon)$ i $2f(x)f'(x) < 0$ en algun interval $(a_i - \eta, a_i)$. Així doncs, ens trobem al cas específic del corol·lari en que $t_i = q_{i-1} - 1$ per ser $f(x)g(x) > 0$ en un interval $(a_i - \epsilon, a_i)$, i $t_i = q_i$ per ser $f(x)g(x) > 0$ en un interval $(a_i, a_i + \epsilon)$. I per tant, tenim $q_{i-1} - 1 = q_i$, és a dir, $q_{i-1} - q_i = 1$ amb $1 \leq i \leq k$, o equivalentment $q_i - q_{i+1} = 1$ per $0 \leq i \leq k - 1$.

La tercera afirmació és evident quan $f(b) \neq 0$, i en el cas que b sí que és una arrel de $f(x)$, aleshores tenim que $b = a_i$ per algun i entre 1 i k i aleshores, $q(S_{f,g}(b)) = t_i = q_{i-1} - 1$. Per tant, $q(S_{f,g}(a)) - q_i$ és el nombre d'arrels a l'interval (a, b) i $q(S_{f,g}(a)) - q_i + 1 = q(S_{f,g}(a)) - q(S_{f,g}(b))$ el nombre d'arrels a l'interval $(a, b]$ □

Observació 3.15. Es pot veure que el teorema anterior cobreix l'original pel Teorema 3.8, mentre que la condició que a i b no siguin arrels múltiples de $f(x)$ no és necessària en aquesta versió del Teorema de Sturm.

4 Implementació en Python

En aquest capítol explicaré els motius pels que he creat una implementació dels conceptes descrits a [1] en Python i faré un resum de la cronologia de la ideació d'aquest programa.

4.1 Objectius

Principalment la idea de fer una implementació en un programa dels resultats del treball sorgeix per poder fer una comprovació numèrica d'aquests amb les proporcions que permeten les eines informàtiques. En segon lloc, el programa és una eina útil per poder calcular la cadena de Sturm i la matriu de Sturm d'un polinomi qualsevol donats els seus coeficients en un temps relativament curt, així com quantificar el nombre d'arrels reals d'aquest polinomi en qualsevol interval.

4.2 Per què Python: l'extensió NumPy

Inicialment, el llenguatge de programació amb el que tenia previst fer aquesta implementació va ser Maple. Tot i això, degut al fet que ja tenia coneixements en Python per l'assignatura Algorísmica, vaig decidir investigar si existia alguna manera de treballar amb polinomis de forma eficient. Aquí apareix NumPy, una extensió de Python que constitueix una biblioteca d'operacions matemàtiques que permet treballar amb vectors i matrius de forma molt més eficient que amb Python estàndard [10]. Així mateix, proporciona eines per poder treballar amb polinomis amb més facilitat. A continuació veiem una comparativa dels temps d'execució d'una operació senzilla amb vectors per l'extensió NumPy i per Python estàndard.

```
import numpy as np
import time
import sys
mida = 10000000
print("vector numpy")
t_vector = time.time()
vector_numpy = np.arange(mida,dtype="int64")
suma_vector = vector_numpy.sum()
print("Tipus d'objecte:"+str(type(vector_numpy)))
print("suma:"+str(suma_vector))
print("Temps d'execució:"+str(time.time()-t_vector))
print("------------")
print("llista")
t_llista = time.time()
llista = list(range(mida))
suma_llista = sum(llista)
print("Tipus d'objecte:"+str(type(llista)))
print("suma:"+str(suma_llista))
print("Temps d'execució:"+str(time.time()-t_llista))
```

```
vector numpy
Tipus d'objecte:<class 'numpy.ndarray'>
suma:49999995000000
Temps d'execució:0.04778575897216797
------------
llista
Tipus d'objecte:<class 'list'>
suma:49999995000000
Temps d'execució:0.42905402183532715
```

Figura 2: Comparativa del temps d'execució per sumar els elements d'un vector

Es pot apreciar que en aquest cas el temps d'execució és al voltant de 10 vegades més

ràpid utilitzant NumPy.

4.3 Funcionament del programa

La funció principal que faig servir de NumPy és la funció Polynomial [9]. Donada una llista de coeficients de mida arbitrària, la funció Polynomial crea una variable que funciona com un polinomi amb els coeficients de la llista de la següent manera:

```
import numpy as np
from numpy.polynomial.polynomial import Polynomial, polyval

coeficients = [1,2,3,4,5]
polinomi = Polynomial(coef = coeficients)
print(polinomi)
```

```
1.0 + 2.0·x¹ + 3.0·x² + 4.0·x³ + 5.0·x⁴
```

Figura 3: El polinomi pren els coeficients d'esquerra a dreta començant pel terme independent i pujant el grau de cada terme en una unitat

NumPy també permet calcular la derivada d'un polinomi amb una funció pròpia així com avaluar el polinomi:

```
import numpy as np
from numpy.polynomial.polynomial import Polynomial, polyval

coeficients = [1,2,3,4,5]
polinomi = Polynomial(coef = coeficients)
print(polinomi)
derivada = polinomi.deriv()
print(derivada)
```

```
1.0 + 2.0·x¹ + 3.0·x² + 4.0·x³ + 5.0·x⁴
2.0 + 6.0·x¹ + 12.0·x² + 20.0·x³
```

Figura 4: El càlcul de la derivada

```
coeficients = [1,2,3,4,5]
polinomi = Polynomial(coef = coeficients)

print(polinomi(-1), polinomi(0), polinomi(1))
```

```
3.0 1.0 15.0
```

Figura 5: El polinomi avaluat en -1, 0 i 1

Clarament, NumPy també te els seus propis vectors i matrius, que funcionen de manera semblant als elements equivalents de Python [10].

Respectant l'ordre cronològic de la construcció del programa, després de comprendre el funcionament de l'extensió NumPy, vaig agafar una variable de tipus Polynomial per poder treballar amb un polinomi concret i posteriorment canviar el programa per poder rebre un polinomi arbitrari. Així doncs, vaig començar amb el polinomi $f(x) = 5x^4 + 4x^3 + 3x^2 + 2x - 1$. Un polinomi de grau 5 amb dues arrels reals, les dues entre -1 i 1, i que per tant, era prou interessant com per fer-lo servir d'exemple. Vaig prendre també una segona variable Polynomial amb la derivada d'aquest primer polinomi.

Observació 4.1. Si bé en la major part de [1] es treballa amb la cadena de Sturm donats dos polinomis $f(x)$, $g(x)$, no considero necessari admetre dos polinomis en el programa, doncs els resultats interessants ocorren en el cas $g(x) = f'(x)$.

El primer que vaig haver d'implementar és la versió modificada de l'algoritme d'Euclides per calcular la cadena de Sturm d'un polinomi. Per fer-ho, ens podem aprofitar del fet que tant la operació // (divisió entera sense residu) com la operació % (mòdul) son compatibles amb la classe Polynomial de NumPy [10].

A pesar de ser senzill, la complicació d'aquest algoritme radica en que al vector de residus apareixen 3 elements després de la primera divisió entera, mentre al vector de quocients n'hi apareix un. És una cosa a tenir en compte a l'hora de crear els bucles, que se soluciona de la següent manera.

```
#Algoritme d'Euclides amb polinprova i la seva derivada (deriv)

quocients = np.array([])              #Creem els vectors de quocients i residus
residus = np.array([polinprova,deriv]) #Al vector de residus afegim f0 i f1
quocients = np.append(quocients, polinprova // deriv)
residus = np.append(residus, - polinprova + quocients[0]*deriv)
i=2                  #Afegim els primers elements dels vectors fora del bucle
                     #perquè els indexos no donin error
while(residus[i]!=poli0): #Compararem sempre amb el polinomi nul en comptes de 0

  quocients = np.append(quocients, residus[i-1] // residus[i])
  residus = np.append(residus, - residus[i-1] % residus[i])
  i=i+1      #Afegim a quocients i a residus les divisions enteres i els residus
             #respectivament fins que l'última divisió hagi tingut residu 0
```

Figura 6: Retall del programa, algoritme d'Euclides modificat (Annex 1)

Tenint la cadena de Sturm (el vector de residus) podem calcular el nombre de variacions en el signe quan aquests polinomis s'avaluen en un valor concret. Aquesta és la funció vfg que calcula aquest nombre sumant el nombre de cops que dos elements consecutius de la cadena tenen signes oposats i sumant també, com es fa notar a la demostració clàssica del teorema de Sturm (Teorema 2.5), el nombre de cops que hi ha un element igual a 0 fora dels extrems de la cadena. Es recorda aquesta propietat en el següent exemple.

Exemple 4.2. Suposem que tenim una cadena de Sturm $f_0(x), \ldots, f_n(x)$ dels polinomis $f_0(x)$, $f_1(x)$ i que calculem el nombre de variacions en $a \in \mathbb{R}$. Si hi ha algun i tal que $f_i(a) = 0$, aleshores tenim:

$$f_{i-1}(a) = d_i f_i(a) - f_{i+1}(a)$$
$$f_{i-1}(a) = -f_{i+1}(a)$$

i per tant hi ha un canvi de signe en la terna $(f_{i+1}(a), f_i(a), f_{i-1}(a))$.

```
def vfg(a): #Definim la funció Vf,g que calcula el nombre de canvis de signe
  j=0       #en la cadena de sturm avaluada en a
  for i in range(len(residus)-2):
    if(residus[i](a)*residus[i+1](a)<0):
      j=j+1 #augmenta j en 1 sempre que hi hagi un canvi de signe consecutiu
    if(residus[i](a)==0) and (i<len(residus)-3) and (i!=0):
      j=j+1 #o un valor igual a 0 fora dels extrems de la cadena
  return(j)
```

Figura 7: Retall del programa, funció vfg (Annex 1)

Exemple 4.3. Comprovem que la funció vfg dona el resultat correcte amb un petit càlcul:

```
#Comprovació de la funcionalitat de vfg
coefaux = list(reversed(coefprova))
rc = np.roots(coefaux)
r = rc[np.isreal(rc)]
n = len(r)
for i in range(n):
  print('Arrel r_i:')
  print(r[i])
  print('Variació de signes en r_i - 0.0001: ',end='')
  print(vfg(r[i]-0.0001))
  print('Variació de signes en r_i + 0.0001: ',end='')
  print(vfg(r[i]+0.0001))
```

```
Arrel r_i:
(-0.9006649757508474+0j)
Variació de signes en r_i - 0.0001: 3
Variació de signes en r_i + 0.0001: 2
Arrel r_i:
(0.29656656293315314+0j)
Variació de signes en r_i - 0.0001: 2
Variació de signes en r_i + 0.0001: 1
```

Figura 8: Càlcul de la variació de signes de la cadena de Sturm al voltant de les arrels del polinomi (Annex 1)

Continuem amb la construcció de la matriu de Sturm, $S_{f,g}(x)$. Tot i la existència de variables matricials a NumPy i si bé és possible tenir una matriu amb polinomis com a coeficients [7], no hi ha una manera de crear aquesta matriu mitjançant un bucle que afegeixi aquests polinomis a la matriu. Així doncs, la matriu $S_{f,g}(x)$ a ulls del programa és realment un vector, tot i això, en podem escriure les files. A continuació hi ha el detall del càlcul de la matriu i com queda escrita pel programa.

Com que no té un format de matriu, no és possible escriure-ho com una quadricula i per tant les files queden desalineades entre elles. De totes maneres, amb aquesta representació es pot visualitzar fàcilment les files de la matriu de Sturm.

```
#Construim S_f,g(x)
mat = np.array([])
for i in range(len(quocients)):
  for j in range(len(quocients)):
    if(i==j):
      mat = np.append(mat, quocients[i])
    elif(i==j-1 or i==j+1):
      mat = np.append(mat, poli1) #Cada element de la matriu serà 0, 1 o d_i(x)
    else:                         #en funció del lloc on es trobi
      mat = np.append(mat, poli0)

#Escrivim la matriu
for i in range(len(quocients)):
  for j in range(len(quocients)): #Modifiquem el final del print per deixar espai a la resta d'elements
    print(mat[i+j*len(quocients)],end='         ')
  print('') #Com que per defecte els prints a Python afegeixen un \n, al final
            #de cada fila fem un print buit perquè canvii de linia
```

```
0.05 + 0.25·x¹          1.0          0.0          0.0
1.0          16.296296296296305 - 22.222222222222225·x¹          1.0          0.0
0.0          1.0          0.029733333333333327 + 0.017999999999999999·x¹          1.0
0.0          0.0          1.0          -25.421724735929395 + 79.81239161280155·x¹
```

Figura 9: Generació de la matriu de Sturm (Annex 1)

Per poder calcular el nombre de valors propis negatius de la matriu de Sturm, calcularem la cadena de Sturm refinada amb una funció senzilla. Per trobar $q(S_{f,g}(x))$ ens ajudarem del Lema 3.1 i la Proposició 3.7 que garanteixen d'una banda que $D_i(x) = \tilde{f}_{m-1}(x)$ i de l'altra, que se satisfà:

$$q(S_{f,g}(x)_{\{m-i+1,\dots,m\}}) =$$
$$= \begin{cases} q(S_{f,g}(x)_{\{m-i+2,\dots,m\}}) & si \quad D_i(x)D_{i-1}(x) > 0 \vee D_i(x) = 0 \\ q(S_{f,g}(x)_{\{m-i+2,\dots,m\}}) + 1 & si \quad D_i(x)D_{i-1}(x) < 0 \vee D_{i-1}(x) = 0 \end{cases}$$

Així doncs, la funció calcula tots els determinants $D_1(x)$ fins a $D_m(x)$ i utilitza aquesta informació per anar trobant els índexs negatius d'inèrcia de cada $S_{f,g}(x)_{\{m-i+1,m-i+2,\dots,m\}}$

```
#Calculem la cadena de Sturm refinada
cadr = np.array([])
for i in range(len(residus)-1):
  cadr = np.append(cadr,residus[i]//residus[-2])
       #fm és el penúltim element del vector residus perquè l'últim és 0



#calculem q(S_f,g(x)) recursivament
def qS_fg(x):
  qS = 0
  m = len(cadr)
  det = np.array([]) #det serà el vector (D_1(x),D_2(x),...,D_m(x))
  det = np.append(det, cadr[-2](x))
  if(det[0]<0):
    qS += 1
  for i in range(2,m):
    det = np.append(det, cadr[-i-1](x))
    if(det[i-1]*det[i-2]<0 or det[i-2]==0): #aqui apliquem les condicions del lema
      qS += 1
  return qS
```

Figura 10: Càlcul de la cadena de Sturm refinada i de l'índex negatiu d'inèrcia de la matriu de Sturm (Annex 1)

Finalment fem una petita comprovació dels valors que retorna la funció vfg i la funció qS_fg a l'interval $[-5, 2]$, on hi trobem les dues arrels reals del polinomi.

```
for i in range(-500,200):
  assert(vfg(i/100)==qS_fg(i/100))
```

Figura 11: Comprovació dels càlculs de les funcions vfg i qS_fg (Annex 1)

En aquest punt tenim acabades les funcions del programa, i queda estructurar-lo de manera que puguem introduir el polinomi que desitgem. Malauradament hi ha una trava i és que Python no té una funció per rebre inputs fora de fitxers o del propi codi del programa [6], per tant la solució serà comentar l'inici del programa amb les instruccions per introduir el polinomi amb el que es voldrà treballar.

```
#Canvia el valor de la variable coefprova pels coeficients del polinomi que
#vulguis analitzar. Per posar els coeficients d'un polinomi a0+a1*x+a2*x^2
#s'escriurà la llista següent: coefprova = [a0,a1,a2]
coefprova = [-1,2,3,4,5]
polinprova = Polynomial(coef = coefprova)
deriv = polinprova.deriv()            #Definim el polinomi f i la seva derivada
cof0 = [0]
```

```
def vfg(a): #Definim la funció Vf,g que calcula el nombre de canvis de signe
  j=0       #en la cadena de sturm avaluada en a
  for i in range(len(residus)-2):
    if(residus[i](a)*residus[i+1](a)<0):
      j=j+1 #augmenta j en 1 sempre que hi hagi un canvi de signe consecutiu
    if(residus[i](a)==0) and (i<len(residus)-3) and (i!=0):
      j=j+1 #o un valor igual a 0 fora dels extrems de la cadena
  return(j)

#Ara es pot cridar la funció vfg i escriure el resultat: si es vol calcular el
#nombre d'arrels reals de polinprova dins l'interval (a,b], s'escriurà el següent:
#print(vfg(a)-vfg(b))
```

Figura 12: Exemples dels comentaris amb les instruccions per utilitzar les funcions del programa (I) (Annex 1)

```
#calculem q(S_f,g(x)) recursivament
def qS_fg(x):
  qS = 0
  m = len(cadr)
  det = np.array([]) #det serà el vector (D_1(x),D_2(x),...,D_m(x))
  det = np.append(det, cadr[-2](x))
  if(det[0]<0):
    qS += 1
  for i in range(2,m):
    det = np.append(det, cadr[-i-1](x))
    if(det[i-1]*det[i-2]<0 or det[i-2]==0): #aqui apliquem les condicions del lema
      qS += 1
  return qS
#Ara, igual que amb la funció vfg, es pot cridar la funció qS_fg i escriure el
#resultat: si es vol calcular el nombre d'arrels reals de polinprova dins
#l'interval (a,b], s'escriurà el següent:
#print(qS_fg(a)-qS_fg(b))
```

Figura 13: Exemples dels comentaris amb les instruccions per utilitzar les funcions del programa (II) (Annex 1)

Per últim, una qüestió interessant que el teorema de Sturm permet calcular de forma ràpida, és el nombre d'arrels reals d'un polinomi. Si fem els límits a $-\infty$ i a $+\infty$ dels polinomis de la cadena de Sturm, sabrem quin és el nombre total d'arrels que el polinomi té a $\mathbb{R}$. Aquest càlcul l'implementem amb uns bucles que adjunten els coeficients de grau més alt de cada polinomi de la cadena de Sturm. D'aquesta manera i sabent si aquests coeficients són d'un terme de grau parell o senar, podem calcular si els límits de cada polinomi de la cadena son positius o negatius i comptar el nombre de variacions de signe.

```
#Càlcul del nombre d'arrels reals d'un polinomi
y = np.array([])
arrelsreals = 0
varinf = 0
varmenysinf = 0 #Mirarem la variació de signes de la cadena de Sturm a -inf i a inf
signes = np.array([])
for i in range(len(residus)):
  x = list(residus[i])
  y = np.append(y, x[-1]) #Per a això construïm un vector amb els coeficients de
  if(len(x)%2==1):        #grau màxim de cada element de la cadena de Sturm
    signes = np.append(signes, 1) #Si aquest coeficient és de grau parell, el multiplicarem
  else:                           #per 1 per mirar la variació a menys infinit
    signes = np.append(signes, -1) #Si el coeficient és imparell el multiplicarem per -1
  print(x[-1])
for i in range(len(y)-1):
  if(y[i]*y[i+1]<0): #La variació de signes a + infinit ve donada pel signe dels coeficients de grau més alt
    varinf += 1
  if(y[i]*y[i+1]*signes[i]*signes[i+1]): #Pel que fa a la variació a - infinit, s'han de tenir en compte si
    varmenysinf += 1                     #els coeficients son de grau parell o imparell
arrelsreals = varmenysinf - varinf
print('El nombre d\'arrels reals del polinomi', end='')
print(polinprova, end=' ')
print('és: ', end='')
print(arrelsreals)
```

Figura 14: Fragment del programa on es calcula el nombre total d'arrels reals del polinomi (Annex 1)

5 Conclusions

Les matemàtiques com a conjunt de ciències tenen sentit en quant a facilitar-ne l'estudi de les seves parts. Sense fer-ho, seria impossible abordar el conjunt del saber matemàtic. Tot i això, en ocasions trobem curioses relacions entre elements diversos dins les matemàtiques. En aquest treball explorem la relació entre un teorema, que en un principi no estava relacionat amb l'àlgebra matricial, i la pròpia teoria de matrius.

En afegit, les eines informàtiques ens permeten fer càlculs i generar elements matemàtics en qüestió de fraccions de segon, i a pesar de la limitació pel que fa a la precisió del càlcul, és converteixen en un recurs especialment útil per posar en pràctica els resultats recollits durant la recerca.

Pel que fa als coneixements que han sigut necessaris per la redacció d'aquest treball, he aprofitat les assignatures d'àlgebra dels primers cursos del grau, així com els coneixements en Python que he adquirit a l'assignatura Algorísmica durant aquest semestre de tardor. Altres competències les hem desenvolupat en paral·lel a la redacció del treball, com per exemple competències en recerca i síntesi, que no havia treballat tant profundament fins al moment. S'ha consultat una diversitat de recursos digitals com llibres, articles, fòrums o vídeos. L'accessibilitat dels recursos bibliogràfics des d'Internet dona lloc al fet que totes les referències provenen de la xarxa, doncs ofereix moltes més possibilitats: eines de cerca, accés instantani des de qualsevol lloc, disponibilitat absoluta de documents...

Això posa un final a aquest projecte. El teorema de Sturm permet acotar les arrels reals d'un polinomi en un nombre finit de passos. És doncs un teorema amb aplicacions interessants sobretot, considero, a nivell estudiantil substituint, o millor, complementant la cerca cega d'arrels de polinomis mitjançant Ruffini. La relació entre la variació de signes de la cadena de Sturm i el nombre de valors propis negatius en la matriu de Sturm dona lloc a preguntar-se quines altres relacions semblants a aquesta poden existir, no obstant, aquesta feina recau sobre investigacions futures.

Referències

[1] Hou, K.; Li, B.: A New Proof of Sturm's Theorem via Matrix Theory, arXiv preprint arXiv:2110.15364., 2021

[2] Yañez, G. El teorema de Sturm, 1983

[3] Travessa, A.; Estructures Algebraiques, 2017

[4] D'Andrea, C.; Cálculo de Raíces Reales de Polinomios, 2006

[5] Muir, T.; Metzler, W. H.; A Treatise on the Theory of Determinants. Courier Corporation, 13:516-525, 2003

[6] Lundhm, F.; Kuchling, A.M.; The Python Standard Library - cap.7.2 Regular expression operations, https://docs.python.org/2/library/re.html#simulating-scanf, 2020

[7] © 2018 Living-Sun.com https://living-sun.com/es/python/703834-matrix-of-polynomial-elements-python-numpy-matrix-sympy.html

[8] EqsQuest Ltd. Symbolab; https://es.symbolab.com/solver/polynomial-calculator/simplificar

[9] Albanez, E.; Polinomios en Python utilizando la clase Polynomial de Numpy, 2021, https://youtu.be/6_okkhLoPRc

[10] Oliphant, T. E.; A guide to NumPy (Vol. 1, p. 85). USA: Trelgol Publishing, 9:p.168, 12:p.212-213, 2006

Annex I

Codi del programa en Python

```
import numpy as np
from numpy.polynomial.polynomial import Polynomial, polyval

#Canvia el valor de la variable coefprova pels coeficients del polinomi que
#vulguis analitzar. Per posar els coeficients d'un polinomi a0+a1*x+a2*x^2
#s'escriurà la llista següent: coefprova = [a0,a1,a2]
coefprova = [-1,2,3,4,5]
polinprova = Polynomial(coef = coefprova)
deriv = polinprova.deriv()             #Definim el polinomi f i la seva derivada
cof0 = [0]
cof1 = [1]
poli0 = Polynomial(coef = cof0)
poli1 = Polynomial(coef = cof1) #Creem també els polinomis p(x)=1 i q(x)=0

#Algoritme d'Euclides amb polinprova i la seva derivada (deriv)

quocients = np.array([])               #Creem els vectors de quocients i residus
residus = np.array([polinprova,deriv]) #Al vector de residus afegim f0 i f1
quocients = np.append(quocients, polinprova // deriv)
residus = np.append(residus, - polinprova + quocients[0]*deriv)
i=2                  #Afegim els primers elements dels vectors fora del bucle
                     #perquè els indexos no donin error
while(residus[i]!=poli0): #Compararem sempre amb el polinomi nul en comptes de 0

  quocients = np.append(quocients, residus[i-1] // residus[i])
  residus = np.append(residus, - residus[i-1] % residus[i])
  i=i+1      #Afegim a quocients i a residus les divisions enteres i els residus
             #respectivament fins que l'última divisió hagi tingut residu 0

#Escrivim el polinomi i la seva cadena de Sturm:
print('f(x)=',end='')
print(polinprova)
print('Cadena de Sturm: [', end='')
for i in range(len(residus)-2):
  print(residus[i], end='')
  print(',', end='')
print(residus[-2], end='')
print(']')

def vfg(a): #Definim la funció Vf,g que calcula el nombre de canvis de signe
  j=0       #en la cadena de sturm avaluada en a
  for i in range(len(residus)-2):
    if(residus[i](a)*residus[i+1](a)<0):
      j=j+1 #augmenta j en 1 sempre que hi hagi un canvi de signe consecutiu
    if(residus[i](a)==0) and (i<len(residus)-3) and (i!=0):
```

```
      j=j+1 #o un valor igual a 0 fora dels extrems de la cadena
  return(j)

#Ara es pot cridar la funció vfg i escriure el resultat: si es vol calcular el
#nombre d'arrels reals de polinprova dins l'interval [a,b], s'escriurà el següent:
#print(vfg(a)-vfg(b))

#Comprovació de la funcionalitat de vfg
'''coefaux = list(reversed(coefprova))
rc = np.roots(coefaux)
r = rc[np.isreal(rc)]
n = len(r)
for i in range(n):
  print('Arrel r_i:')
  print(r[i])
  print('Variació de signes en r_i - 0.0001: ',end='')
  print(vfg(r[i]-0.0001))
  print('Variació de signes en r_i + 0.0001: ',end='')
  print(vfg(r[i]+0.0001))'''

#Construim S_f,g(x)
mat = np.array([])
for i in range(len(quocients)):
  for j in range(len(quocients)):
    if(i==j):
      mat = np.append(mat, quocients[i])
    elif(i==j-1 or i==j+1):
      mat = np.append(mat, poli1) #Cada element de la matriu serà 0, 1 o d_i(x)
    else:                         #en funció del lloc on es trobi
      mat = np.append(mat, poli0)

#Escrivim la matriu
print('Matriu de Sturm:')
for i in range(len(quocients)):
  for j in range(len(quocients)): #Modifiquem el final del print per deixar espai a la re
    print(mat[i+j*len(quocients)],end='         ')
  print('') #Com que per defecte els prints a Python afegeixen un \n, al final
            #de cada fila fem un print buit perquè canvii de linia

#Calculem la cadena de Sturm refinada
cadr = np.array([])
for i in range(len(residus)-1):
  cadr = np.append(cadr,residus[i]//residus[-2])
       #fm és el penúltim element del vector residus perquè l'últim és 0

#Escrivim la cadena de Sturm refinada
print('Cadena de Sturm refinada: [', end='')
for i in range(len(cadr)-1):
  print(cadr[i], end='')
```

```
  print(',', end='')
print(cadr[-1], end='')
print(']')

#calculem q(S_f,g(x)) recursivament
def qS_fg(x):
  qS = 0
  m = len(cadr)
  det = np.array([]) #det serà el vector (D_1(x),D_2(x),...,D_m(x))
  det = np.append(det, cadr[-2](x))
  if(det[0]<0):
    qS += 1
  for i in range(2,m):
    det = np.append(det, cadr[-i-1](x))
    if(det[i-1]*det[i-2]<0 or det[i-2]==0): #aqui apliquem les condicions del lema
      qS += 1
  return qS
#Ara, igual que amb la funció vfg, es pot cridar la funció qS_fg i escriure el
#resultat: si es vol calcular el nombre d'arrels reals de polinprova dins
#l'interval [a,b], s'escriurà el següent:
#print(qS_fg(a)-qS_fg(b))

for i in range(-500,200):
  assert(vfg(i/100)==qS_fg(i/100))

#Càlcul del nombre d'arrels reals d'un polinomi
y = np.array([])
arrelsreals = 0
varinf = 0
varmenysinf = 0 #Mirarem la variació de signes de la cadena de Sturm a -inf i a inf
signes = np.array([])
for i in range(len(residus)):
  x = list(residus[i])
  y = np.append(y, x[-1]) #Per a això construïm un vector amb els coeficients de
  if(len(x)%2==1):        #grau màxim de cada element de la cadena de Sturm
    signes = np.append(signes, 1) #Si aquest coeficient és de grau parell, el multiplicar
  else:                           #per 1 per mirar la variació a menys infinit
    signes = np.append(signes, -1) #Si el coeficient és imparell el multiplicarem per -1
  print(x[-1])
for i in range(len(y)-1):
  if(y[i]*y[i+1]<0): #La variació de signes a + infinit ve donada pel signe dels coeficie
    varinf += 1
  if(y[i]*y[i+1]*signes[i]*signes[i+1]): #Pel que fa a la variació a - infinit, s'han de
    varmenysinf += 1                     #els coeficients son de grau parell o imparell
arrelsreals = varmenysinf - varinf
print('El nombre d\'arrels reals del polinomi', end='')
print(polinprova, end=' ')
print('és: ', end='')
print(arrelsreals)
```