עיבוד אותות אקראיים

פרויקט קורס

Comparison of Autoregressive estimation methods by least square

מרצה: ד"ר יצחק לפידות

מגיש : יזיד בישאראת 315914689

Table of Contents

רקע כללי 3

מודל אוטו-רגרסיבי (Autoregressive model) 3

בעיית אוטו רגרסיה 3

שיטת Yule-walker(YW): 4

רקורסית לוינסון (Levinson Recursion) 5

שיטת פרוני (Prony's Method) 7

Direct inversion 9

התאמת מודל פולינומי 12

הסבר תהליך הניסוי 12

תוצאות 15

שיטות שערוך מסנן 16

הסבר תהליך הניסוי 16

תוצאות סדר משתנה 18

תוצאות עם שינוי אורך אות הרעש 20

מסקנות וסיכום 22

קוד 23

References 26

רקע כללי

מודל אוטו-רגרסיבי (Autoregressive model)

מודל אוטו-רגרסיבי(AR) זה מודל בו המוצא הוא תלוי בערכי המוצא הקודמים, מודל AR הוא מקרה פרטי למשוואת ההפרשים:

$$\sum_{k=0}^{N} a_k y(n-k) = \sum_{k=0}^{M} b_k x(n-k)$$

מודל AR תלוי רק בערכי העבר לכן:

$$\sum_{k=0}^{N} a_k y(n-k) = x(n)$$

אם מבצעים התמרת Z על מודל AR נקבל:

$$Y(z)\sum_{k=0}^{N} a_k z^{-k} = X(z)$$

בעיית אוטו רגרסיה

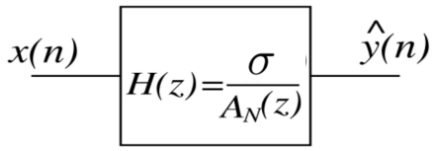
כשמסכלים על המודל AR, אפשר להתיחס אליו כמשסנן IIR כלומר יש רק קטבים חוץ מאפס אחד באפס:

$$H(z) = \frac{\sigma}{A(z)}.$$

כשאר:

$$A(z)=\sum_{k=0}^{N} a_k z^{-k}$$

מבנה המערכת:



אז המטרה של הפרויקט היא לבנות מודל לבעייה כאשר המוצא של המודל שאשבנה יהיה מאוד קרוב למודל של המשוואה המקורית, אז אנחנו צריכים לשחק במקדמי הפולינום A כדי להקטין את השגיאה בין המוצא המקורי למודל שבונים.

שיטת Yule-walker(YW):

אז המודל הוא :

$$H(z) = \frac{\sigma}{A(z)}.$$

ומשוואת ההפרשים של AR היא:

$$y(n) + a_1 y(n-1) + \cdots + a_N y(n-N) = x(n)$$

$$a_0 = 1$$

נניח גם שהאות סתציונרי :

$$\mathrm{E}\{x(n)x(n-k)\} = \sigma^2 \delta(k)$$

ומודל AR נותן:

$$y(n) = x(n) + \alpha_1 x(n-1) + \alpha_2 x(n-2) + \cdots$$

אם נעשה את זה לכל הדימות נקבל פונקצית מדריגה מתחילה ב אפס:

$$\mathrm{E}\{y(n)x(n)\} = \mathrm{E}\{[x(n) + \alpha_1 x(n-1) + \alpha_2 x(n-2) + \cdots]x(n)\} = \sigma^2$$

באופו דומה לכל מרווח בין דגימות :

$$\begin{aligned}\mathrm{E}\{y(n-k)x(n)\} &= \mathrm{E}\{[x(n-k) + \alpha_1 x(n-k-1) + \cdots]\, x(n)\} \\ &= 0 \quad \text{for } k > 0\end{aligned}$$

בהצגת משוואת הAR בצורה וקטורית נקבל :

$$[y(n), y(n-1), \ldots, y(n-N)] \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = x(n)$$

$$\begin{aligned}\mathrm{E}\{y(n)x(n)\} &= \mathrm{E}\left\{y(n)\, [y(n), y(n-1), \ldots, y(n-N)] \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{bmatrix}\right\} \\ &= [r_0, r_1, \ldots, r_N] \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \sigma^2.\end{aligned}$$

$$\mathrm{E}\{y(n-k)x(n)\} = \mathrm{E}\left\{ y(n-k)\,[y(n),\ldots,y(n-N)] \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} \right\}$$

$$= [r_k, r_{k-1}, \ldots, r_{k-N}] \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = 0 \quad \text{for } k > 0.$$

$$\begin{bmatrix} r_0 & r_1 & \cdots & r_N \\ r_1 & r_0 & \cdots & r_{N-1} \\ \vdots & \vdots & \vdots & \vdots \\ r_N & r_{N-1} & \cdots & r_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_1 \\ \vdots \\ a_N \end{bmatrix} = \begin{bmatrix} \sigma^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

אם נשאיר את המשוואה הראשונה בחוץ נקבל:

$$\begin{bmatrix} r_0 & r_1 & \cdots & r_{N-1} \\ r_1 & r_0 & \cdots & r_{N-2} \\ \vdots & \vdots & \vdots & \vdots \\ r_{N-1} & r_{N-2} & \cdots & r_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} = - \begin{bmatrix} r_1 \\ r_2 \\ \vdots \\ r_N \end{bmatrix}$$

בצורה מטריצית:

$$R\boldsymbol{a} = -\boldsymbol{r}$$

משואות אלו נקראות משואות Yule-Walker equations

רקורסית לוינסון (Levinson Recursion)

השיטה הזות ידועה כי רקורסית לוינסון, הינה שיטה לחישוב פתרון מערכת משוואות כשאר מערכת המשוואת הינה טופליצית, סיבוכיות האלגוריתם הינו $\Theta(n^2)$ שזהו שיפור משמעותי לעומת דירוג גאוס (דירוג מטריצות) בעל הסיבוכיות $\Theta(n^3)$.

- נשנה את הסימנים של משוואת יול ווקר כדי שיתאימו המשוואה החדשה:

$$\begin{bmatrix} r_0 & r_1 & \cdots & r_N \\ r_1 & r_0 & \cdots & r_{N-1} \\ \vdots & \vdots & \vdots & \vdots \\ r_N & r_{N-1} & \cdots & r_0 \end{bmatrix} \begin{bmatrix} 1 \\ a_{N,1} \\ \vdots \\ a_{N,N} \end{bmatrix} = \begin{bmatrix} \sigma_N^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$R_N \boldsymbol{a}_N = \sigma_N^2 \boldsymbol{e}_1$$

כאשר

$$\boldsymbol{e}_1 = [1, 0, 0, \ldots, 0]^T$$

עבור N=1,

$$\begin{aligned} r_0 + r_1 a_{1,1} &= \sigma_1^2, \\ r_1 + r_0 a_{1,1} &= 0, \end{aligned}$$

לכן

$$
\begin{aligned}
a_{1,1} &= -\frac{r_1}{r_0}, \\
\sigma_1^2 &= r_0\left\{1-\left[\frac{r_1}{r_0}\right]^2\right\}.
\end{aligned}
$$

המטרה: זה כשנתון a_N אנחנו רוצים למצוא את הפתרון לערכת מסדר (N+1) הזות:

$$R_{N+1}\boldsymbol{a}_{N+1} = \sigma_{N+1}^2\boldsymbol{e}_1$$

נצרף 0 לווקטור a_N ונכפיל הוקטור ב- $RN+1$:

$$
\begin{bmatrix}
r_0 & r_1 & \cdots & r_N & r_{N+1} \\
r_1 & r_0 & \cdots & r_{N-1} & r_N \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
r_N & r_{N-1} & \cdots & r_0 & r_1 \\
r_{N+1} & r_N & \cdots & r_1 & r_0
\end{bmatrix}
\begin{bmatrix}
1 \\ a_{N,1} \\ \vdots \\ a_{N,N} \\ 0
\end{bmatrix}
=
\begin{bmatrix}
\sigma_N^2 \\ 0 \\ \vdots \\ 0 \\ \gamma_N
\end{bmatrix}
$$

$$\text{where} \quad \gamma_N = r_{N+1} + \sum_{k=1}^{N} a_{N,k} r_{N+1-k}.$$

נשתמש בתכונת הסימטריות טופליצית ל$RN+1$ כדי להציג בצורה הבא:

$$
\begin{bmatrix}
r_0 & r_1 & \cdots & r_N & r_{N+1} \\
r_1 & r_0 & \cdots & r_{N-1} & r_N \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
r_N & r_{N-1} & \cdots & r_0 & r_1 \\
r_{N+1} & r_N & \cdots & r_1 & r_0
\end{bmatrix}
\begin{bmatrix}
0 \\ a_{N,N} \\ \vdots \\ a_{N,1} \\ 1
\end{bmatrix}
=
\begin{bmatrix}
\gamma_N \\ 0 \\ \vdots \\ 0 \\ \sigma_N^2
\end{bmatrix}
$$

כעת נבצע סכימה ממושקלת של שני המשוואות לעיל:

$$
R_{N+1}\left\{
\begin{bmatrix}
1 \\ a_{N,1} \\ \vdots \\ a_{N,N} \\ 0
\end{bmatrix}
+\Gamma_{N+1}
\begin{bmatrix}
0 \\ a_{N,N} \\ \vdots \\ a_{N,1} \\ 1
\end{bmatrix}
\right\}
=
\begin{bmatrix}
\sigma_N^2 \\ 0 \\ \vdots \\ 0 \\ \gamma_N
\end{bmatrix}
+\Gamma_{N+1}
\begin{bmatrix}
\gamma_N \\ 0 \\ \vdots \\ 0 \\ \sigma_N^2
\end{bmatrix}
$$

ונבחר:

$$\Gamma_{N+1} = -\frac{\gamma_N}{\sigma_N^2}$$

מה שמצמצם את המשוואה לעיל ל

$$R_{N+1}\boldsymbol{a}_{N+1} = \sigma_{N+1}^2\boldsymbol{e}_1$$

כאשר

$$
\boldsymbol{a}_{N+1} =
\begin{bmatrix}
1 \\ a_{N,1} \\ \vdots \\ a_{N,N} \\ 0
\end{bmatrix}
+\Gamma_{N+1}
\begin{bmatrix}
0 \\ a_{N,N} \\ \vdots \\ a_{N,1} \\ 1
\end{bmatrix}
\quad \text{and} \quad
\sigma_{N+1}^2 = \sigma_N^2 + \Gamma_{N+1}\gamma_N = \sigma_N^2[1-\Gamma_{N+1}^2]
$$

שיטת פרוני(Prony's Method)

שיטה זו הומצאה בסוף המאה ה- 18 שימושה מיועד לחישובים דיגיטליים של המחשב.
בדומה להתמרת פורייה שיטת פרוני מחלצת מידע בעל ערך מאות אשר נדגם בתדר דגימה אחיד ובונה סדרה של סינוסואידים או מעריכים מרוכבים. הדבר מאפשר לשערך תדירות, אמפליטודה, פאזה ומרכיבי דעיכה של האות.
שיטת פרוני היא למעשה פירוק האות ל M -מעריכים מרוכבים הניתנים למציאה ע"י התהליך הבא:

$\hat{f}(t)$ מייצגת את הדגימה ה n -מתוך N וניתנת לרישום באופן הבא:

$$F_n = \hat{f}(\Delta_t n) = \sum_{m=1}^{M} \mathrm{B}_m e^{\lambda_m \Delta_t n}, \quad n = 0, \ldots, N-1.$$

מכיוון ש $\hat{f}(t)$ מורכבת ע"י סינוסואידים ניתן לפרק לזוגות של מעריכים מרוכבים באופן הבא:

$$\begin{aligned}
\mathrm{B}_a &= \frac{1}{2} A_i e^{\phi_i j}, \\
\mathrm{B}_b &= \frac{1}{2} A_i e^{-\phi_i j}, \\
\lambda_a &= \sigma_i + j\omega_i, \\
\lambda_b &= \sigma_i - j\omega_i,
\end{aligned}$$

כאשר:

$$\begin{aligned}
\mathrm{B}_a e^{\lambda_a t} + \mathrm{B}_b e^{\lambda_b t} &= \frac{1}{2} A_i e^{\phi_i j} e^{(\sigma_i + j\omega_i)t} + \frac{1}{2} A_i e^{-\phi_i j} e^{(\sigma_i - j\omega_i)t} \\
&= A_i e^{\sigma_i t} \cos(\omega_i t + \phi_i).
\end{aligned}$$

בגלל שהסכימה של המעריכים המרוכבים היא פתרון הומוגני למשוואת ההפרשים הלינארית, משוואת ההפרשים הבא מתקיים:

$$\hat{f}(\Delta_t n) = -\sum_{m=1}^{M} \hat{f}[\Delta_t (n-m)] P_m, \quad n = M, \ldots, N-1.$$

המפתח בשיטת פרוני הוא שהמקדמים של משוואת ההפרשים קשורים לפולינום הבא:

$$z^M + P_1 z^{M-1} + \cdots + P_M = \prod_{m=1}^{M} \left(z - e^{\lambda_m}\right).$$

עובדה זו מובילה לפתרון שיטת פרוני בשלושה השלבים הבאים:

1. בנייה ופתירה של המערכת הבאה עבור ערכי Pm :

$$\begin{bmatrix} F_M \\ \vdots \\ F_{N-1} \end{bmatrix} = \begin{bmatrix} F_{M-1} & \cdots & F_0 \\ \vdots & \ddots & \vdots \\ F_{N-2} & \cdots & F_{N-M-1} \end{bmatrix} \begin{bmatrix} P_1 \\ \vdots \\ P_M \end{bmatrix}.$$

2. לאחר מציאת ערכי Pm יש למצוא את השורשים, בדרך נומרית במידת הצורך, של הפולינוף :

$$z^M + P_1 z^{M-1} + \cdots + P_M,$$

שורש ה m -של הפולינום יהיה שווה ל e^{λ_m}.

3. ע"י מציאת ערכי e^{λ_m} ערכי Fn הם חלק ממערכת המשוואות הלינארית אשר על ידי ניתן למצוא מעריך Bm:

$$\begin{bmatrix} F_{k_1} \\ \vdots \\ F_{k_M} \end{bmatrix} = \begin{bmatrix} (e^{\lambda_1})^{k_1} & \dots & (e^{\lambda_M})^{k_1} \\ \vdots & \ddots & \vdots \\ (e^{\lambda_1})^{k_M} & \dots & (e^{\lambda_M})^{k_M} \end{bmatrix} \begin{bmatrix} \mathrm{B}_1 \\ \vdots \\ \mathrm{B}_M \end{bmatrix}$$

כאשר ישנם M ערכים עבורki .

כעת נראה כיצד ניתן לפתור בעזרת שיטת פרוני בעיית אוטו-רגרסיה. נניח את משוואת אוטו-רגרסיה:

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + x(n)$$

נכתוב עבור L-N נקודות שנמדדו $\{y(n)\}_N^{L-1}$ בצורה מטריצית:

$$\begin{bmatrix} y(N) \\ y(N+1) \\ \vdots \\ y(L-1) \end{bmatrix} = -\begin{bmatrix} y(N-1) & y(N-2) & \cdots & y(0) \\ y(N) & y(N-1) & \cdots & y(1) \\ \vdots & \vdots & \vdots & \vdots \\ y(L-2) & & \cdots & y(L-N-1) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_N \end{bmatrix} + \begin{bmatrix} x(N) \\ x(N+1) \\ \vdots \\ x(L-1) \end{bmatrix}$$

ובייצוג מטריצי: $y = -Ya + x$

למציאת הפתרון נשתמש ב LS לדוגמא לצמצם את:

$$\|\boldsymbol{x}\|^2 = (\boldsymbol{y} + Y\boldsymbol{a})^H (\boldsymbol{y} + Y\boldsymbol{a})$$

הפתרון נתון ע"י המשוואות הנורמליות – משוואות אלו הן משוואות אשר מקטינות את סכום ריבועי ההפרשים בין צד שמאל וצד ימין.

$$Y^H Y \boldsymbol{a} = -Y^H \boldsymbol{y}$$

ע"י פתירת המשוואות הנורמליות אנו מקבלים את a. בצורה פורמלית ניתן לכתוב את הפתרון כך:

$$\boldsymbol{a} = -(Y^H Y)^{-1} Y^H \boldsymbol{y}$$

ישנה אנלוגיה בין משוואות יול-ווקר ובין המשוואות הנורמליות(פרוני)

$$R \leftrightarrow Y^H Y \qquad r \leftrightarrow Y^H \boldsymbol{y}. \qquad R\boldsymbol{a} = -\boldsymbol{r}, \qquad Y^H Y \boldsymbol{a} = -Y^H \boldsymbol{y},$$

פרקטית אניו יכולים לשערך את מטריצת האוטו קורלציה המדויקת R ואת וקטור האוטו-קורלציה.

$$\widehat{R} = Y^H Y \qquad \widehat{r} = Y^H \boldsymbol{y}$$

Direct inversion

נציג את הדרך האנליטית בה נפתור את המשוואות הנורמליות של שיטת פרוני ע"י היפוך מטריצה ישיר $x_{i+1} = \phi_1 x_i + \xi_{i+1}$

התאמת הסימונים לעומת הסימונים בפיתוחים האחרים:

כניסה נוכחית -ξi+ , מקדם מספר P- ϕp , יציאה נוכחית -xi+1

ולכן משוואת ההפרשים בפיתוח הנוכחי ניראת כך–

$$x_{i+1} = \phi_1 x_i + \phi_2 x_{i-1} + \cdots + \phi_p x_{i-p+1} + \xi_{i+1}$$

<u>פתרון עבור מקדם בודד P = 1:</u>

נציג את המערכת בצורהOver-detemined system :

$$\underbrace{\begin{pmatrix} x_2 \\ x_3 \\ \vdots \\ x_N \end{pmatrix}}_{\mathbf{b}} = \underbrace{\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_{N-1} \end{pmatrix}}_{\mathbf{A}} \phi_1$$

כאשר מערכת כזו נפתרת ללא כל קושי בעזרת משערך **LS** באופן הבא:

$$\hat{\phi}_1 = \left(\mathbf{A}^T \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{b} = \frac{\sum_{i=1}^{N-1} x_i x_{i+1}}{\sum_{i=1}^{N-1} x_i^2} = \frac{c_1}{c_o} = r_1$$

כאשר ci,ri הם מקדמי האוטו-קוריאנס והאוטו-קורולציה במקום ה **i** -בהתאמה.

<u>פתרון עבור מקדם בודד – P=2:</u>

כעת הבעיה נראת כך:

$$x_{i+1} = \phi_1 x_i + \phi_2 x_{i-1} + \xi_{i+1}$$

נציג את המערכת בצורהOver-detemined system :

לעומת הפתרון עבור מקדם בודד לבעיה הנתונה כעת
הביטוי אשר נפתח לפתרון אנליטי, בשימוש משערך **LS** אינו טריוויאלי:

$$\underbrace{\begin{pmatrix} x_3 \\ x_4 \\ \vdots \\ x_N \end{pmatrix}}_{\mathbf{b}} = \underbrace{\begin{pmatrix} x_2 & x_1 \\ x_3 & x_2 \\ \vdots & \vdots \\ x_{N-1} & x_{N-2} \end{pmatrix}}_{\mathbf{A}} \underbrace{\begin{pmatrix} \phi_1 \\ \phi_2 \end{pmatrix}}_{\mathbf{\Phi}}$$

$$\hat{\mathbf{\Phi}} = \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b}$$

נתחיל בפיתוח-

$$\left(\mathbf{A}^T\mathbf{A}\right)^{-1} = \left[\begin{pmatrix} x_2 & x_3 & \cdots & x_{N-1} \\ x_1 & x_2 & \cdots & x_{N-2} \end{pmatrix}\begin{pmatrix} x_2 & x_1 \\ x_3 & x_2 \\ x_{N-1} & x_{N-2} \end{pmatrix}\right]^{-1}$$

$$= \begin{pmatrix} \sum_{i=2}^{N-1} x_i^2 & \sum_{i=2}^{N-1} x_i x_{i-1} \\ \sum_{i=2}^{N-1} x_i x_{i-1} & \sum_{i=1}^{N-2} x_i^2 \end{pmatrix}^{-1}$$

$$= \frac{1}{\sum_{i=2}^{N-1} x_i^2 \sum_{i=1}^{N-2} x_i^2 - \sum_{i=2}^{N-1} x_i x_{i-1} \sum_{i=2}^{N-1} x_i x_{i-1}} \begin{pmatrix} \sum_{i=1}^{N-2} x_i^2 & -\sum_{i=2}^{N-1} x_i x_{i-1} \\ -\sum_{i=2}^{N-1} x_i x_{i-1} & \sum_{i=2}^{N-1} x_i^2 \end{pmatrix}$$

עתה נשתמש בעובדה שהסדרה שבידינו **xi** הינה סטציונרית, ולכן אלמנטי האוטו-קוריאנס הינם פונקציה של הפרש הזמנים בלבד. במקרה זה נקבל:

$$\left(\mathbf{A}^T\mathbf{A}\right)^{-1} = \frac{1}{c_o^2 - c_1^2}\begin{pmatrix} c_o & -c_1 \\ -c_1 & c_o \end{pmatrix},$$

$$\left(\mathbf{A}^T\mathbf{A}\right)^{-1} = \frac{1}{c_o^2(1 - r_1^2)}\begin{pmatrix} c_o & -c_1 \\ -c_1 & c_o \end{pmatrix},$$

$$\left(\mathbf{A}^T\mathbf{A}\right)^{-1} = \frac{1}{c_o(1 - r_1^2)}\begin{pmatrix} r_o & -r_1 \\ -r_1 & r_o \end{pmatrix}.$$

באופן דומה:

$$\mathbf{A}^T\mathbf{b} = \begin{pmatrix} x_2 & x_3 & \cdots & x_{N-1} \\ x_1 & x_2 & \cdots & x_{N-2} \end{pmatrix}\begin{pmatrix} x_3 \\ x_4 \\ \vdots \\ x_N \end{pmatrix} = \begin{pmatrix} \sum_{i=3}^{N} x_i x_{i-1} \\ \sum_{i=3}^{N} x_i x_{i-2}, \end{pmatrix}$$

וגם כאן נשתמש בעובדה השסדרה שבידינו הינה סטציונרית ונקבל : $\mathbf{A}^T\mathbf{b} = \begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$

שילוב של שני הביטויים שפיתחנו יתנו לנו:

$$\left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\mathbf{b} = \frac{1}{c_o(1 - r_1^2)}\begin{pmatrix} r_o & -r_1 \\ -r_1 & r_o \end{pmatrix}\begin{pmatrix} c_1 \\ c_2 \end{pmatrix}$$

$$= \frac{1}{1 - r_1^2}\begin{pmatrix} 1 & -r_1 \\ -r_1 & 1 \end{pmatrix}\begin{pmatrix} r_1 \\ r_2 \end{pmatrix}.$$

ואם נפרק את הייצוג לייצוג של כל מקדם בנפרד נקבל:

$$\hat{\phi}_1 = \frac{r_1\left(1-r_2\right)}{1-r_1^2} \quad \hat{\phi}_2 = \frac{r_2-r_1^2}{1-r_1^2}$$

כמובן שישנה אפשרות להמשיך ולפתח גם עבור $p\geq 3$ באותו אופן, אך האלגברה, שכבר בפיתוח לשני מקדמים ראינו שאינה בסיסית, הופכת לעמוסה ביותר, לדוגמא נניח שלושה מקדמים:

$$\mathbf{A}^T\mathbf{A} = \begin{pmatrix} c_o & c_1 & c_2 \\ c_1 & c_o & c_1 \\ c_2 & c_1 & c_o \end{pmatrix}$$

הדטרמיננטה של המטריצה הכרחית לחישוב המטריצה ההפוכה ניראת כה מסורבלת:

$$\det\left(\mathbf{A}^T\mathbf{A}\right) = c_o\left(c_o^2 - 2c_1^2 + 2\frac{c_1^2c_2}{c_o} - c_2^2\right) = c_o\left[c_o^2 + 2c_1^2\left(r_2-1\right) - c_2^2\right]$$

וכאשר אנו מכפילים אותה במטריצה הנוספת מתקבלים ביטויים ארוכים מאד. ישנם דרכים פשוטות יותר ושיטות טובות יותר לקבלת מקדמים של בעיית אוטו-רגרסיה, כפי שראינו אלגוריתם לווינסון-דארבין הפותר את משוואות לוינסון דארבין.

התאמת מודל פולינומי

הסבר תהליך הניסוי

בחלק זה איממש תהליך שנקרא polynomial regression באמצעות שיטת , least-square התהליך נקרא גם. polynomial least squares fittings
הרעיון הוא למצוא פונקציית פולינום שתתאים כראוי לסט נתון של נקודות , בעוד שקביעת סדר המודל (דרגת הפולינום) נתונה בידינו.
התאמה(fitting) דורשת מודל פרמטרי שקושר את נתוני התגובה לנתוני המבוא בעזרת מקדם אחד או יותר. תוצאת תהליך ההתאמה הינה שערוך של מקדמי המודל.
בשערוך מקדמים בשיטת LS מנשה להביא למינימום את הסכום הריבועי של השאריות, כאשר השארית(residual) של איטרציה i מוגדרת כהפרש בין ערך התגובה המקורי לבין ערך התגובה המשוערך–

כאשר

$$r_i = y_i - \hat{y}_i$$

residual=data – fit

The summed square of residuals is given by

$$S = \sum_{i=1}^{n} r_i^2 = \sum_{i=1}^{n} \left(y_i - \hat{y}_i\right)^2$$

כאשר n הוא מספר הנקודות בסט המידע הנתון.

כעת אתמקד בשיטה linear least squares שמאפשרת התאמה למודלים לינאריים כמו פולינומים. אסמן את סדר המודל ב m -ואקבל את הפיתוח הבא–

$$\hat{y} = b_1 * x^m + b_2 * x^{m-1} + \cdots + b_{m+1}$$

$$S = \sum^{n} (y_i - \hat{y})^2 = \sum^{n} (y_i - (b_1 * x_i^m + b_2 * x_i^{m-1} + \cdots + b_{m+1}))^2$$

$$\frac{dS}{db_1} = -2 * \sum_{i=1}^{n} x_i^m (y_i - \hat{y})^1 = 0 \ ,\ldots\ , \qquad \frac{dS}{db_{m+1}} = -2 * \sum_{i=1}^{n} (y_i - \hat{y})^1 = 0 \quad \rightarrow$$

$$\sum_{i=1}^{n} x_i^m (y_i - \hat{y})^1 = 0 \ ,\ldots\ , \ \sum_{i=1}^{n} (y_i - \hat{y})^1 = 0 \ \rightarrow$$

$$\sum_{i=1}^{n} x_i^m (y_i - (b_1 * x_i^m + b_2 * x_i^{m-1} + \cdots + b_{m+1})\ ) = 0 \ ,\ldots$$

$$\ldots\ , \ \sum_{i=1}^{n} (y_i - (b_1 * x_i^m + b_2 * x_i^{m-1} + \cdots + b_{m+1})\ ) = 0 \ \rightarrow$$

המשוואה הנורמלית—

$$b_1\sum_{i=1}^{n} x_i^{2m} + b_2\sum_{i=1}^{n} x_i^{2m-1} + \cdots + b_{m+1}\sum_{i=1}^{n} x_i^{m} = \sum_{i=1}^{n} x_i^{m} y_i \quad , \ldots$$

$$b_1\sum_{i=1}^{n} x_i^{m} + b_2\sum_{i=1}^{n} x_i^{m-1} + \cdots + b_{m+1} = \sum_{i=1}^{n} y_i$$

בצורה מטריצית, מודילים לינארים על ידי הנוסחה:

$$\boldsymbol{y}=\mathbf{X}*\mathbf{b}$$

כאשר y וקטור התגובה, b וקטור המקדמים

המשוואות הנורמליות מתקבלות על ידי הכפלת שני הצדדים ב-X^T

$$\boldsymbol{X}^T * \boldsymbol{y} = (\boldsymbol{X}^T\mathbf{X}) * \mathbf{b} \rightarrow \quad \boldsymbol{b} = (\boldsymbol{X}^T\mathbf{X})^{-1} * \boldsymbol{X}^T * \boldsymbol{y}$$

ההכפלה בX^T הינה הכרחית כדי לקבל מטריצה ריבועית X^TX שלה ניתן למצוא הופכית, שהרי מטריצה X היא לא בהכרח ריבועית ולכן אינה הפיכה.

הפתרון הנ"ל נקרא Direct inversion, ובו המקדם מחושבים על ישי היפוך של מטריצת האוטו- קורלציה באופן ישיר(אנליטי).

להלן תיאור הפונקציה שכתבנו למימוש ופתרון מערכת המשוואות בצורה מטריצית:

```
function [b] = My_LsPolyfit(x,y,m)
% MyLsPolyfit(x, y, m) fits a least-squares polynomial of degree m .
% The output is the row vector b of coefficients [b1;b2;..;bm+1] .
% b is in descending powers Pm(x) = b1*x^m + b2*x^m-1 +...+ bm+1 .
```

וכן חישוב המוצא—

```
% y = X*b -> X'*y = X'X*b -> b = inv(X'X*)*X'*y
b = fliplr( inv(X'*X)*X'*y );
```

כעת אשתמש בפונקציה שכתבנו ונבצע שערוך מודל פולינום, עבור מידע סופי נתון, בנוסף אשווה את התוצאות עם שערוך של פונקציה מובנית של מטלב polyfit . ההשוואה נעשית עבור דרגות פולינום משתנות.

תיאור הפונקציה Polyfit הינו–

p = polyfit (x, y, n) finds the coefficients of a polynomial p(x) of degree n that fits the data y best in a **least-squares sense**.
p is a row vector of length n + 1 containing the polynomial coefficients in descending powers.

אפשר לראות שגם פונקציה זו משתמשת ברעיון שלLS , כך שמההשוואה נוכל לקבל מושג על דיוק ונכונות הפונקציה שלנו.

הנתונים הם–

```
% data coordinates
x = [1 2 3 4 5.5 7 10];
y = [3 7 9 15 22 21 21];
```

נתונים אלו, בנוסף לסדר המודל, מוכנסים לפונקציות ההתאמה—

```
% calculate coefficients
matlab_coeff = polyfit(x, y, order);
my_coeff = MyLsPolyfit(x, y, order);
```

השתמשתי בפונקציה מובנית של מטלב polyval(p,x) שמחשבת את ערך הפולינום שווקטור מקדמיו הוא p בנקודה מסוימת x .

להלן קטע הקוד המתאים–

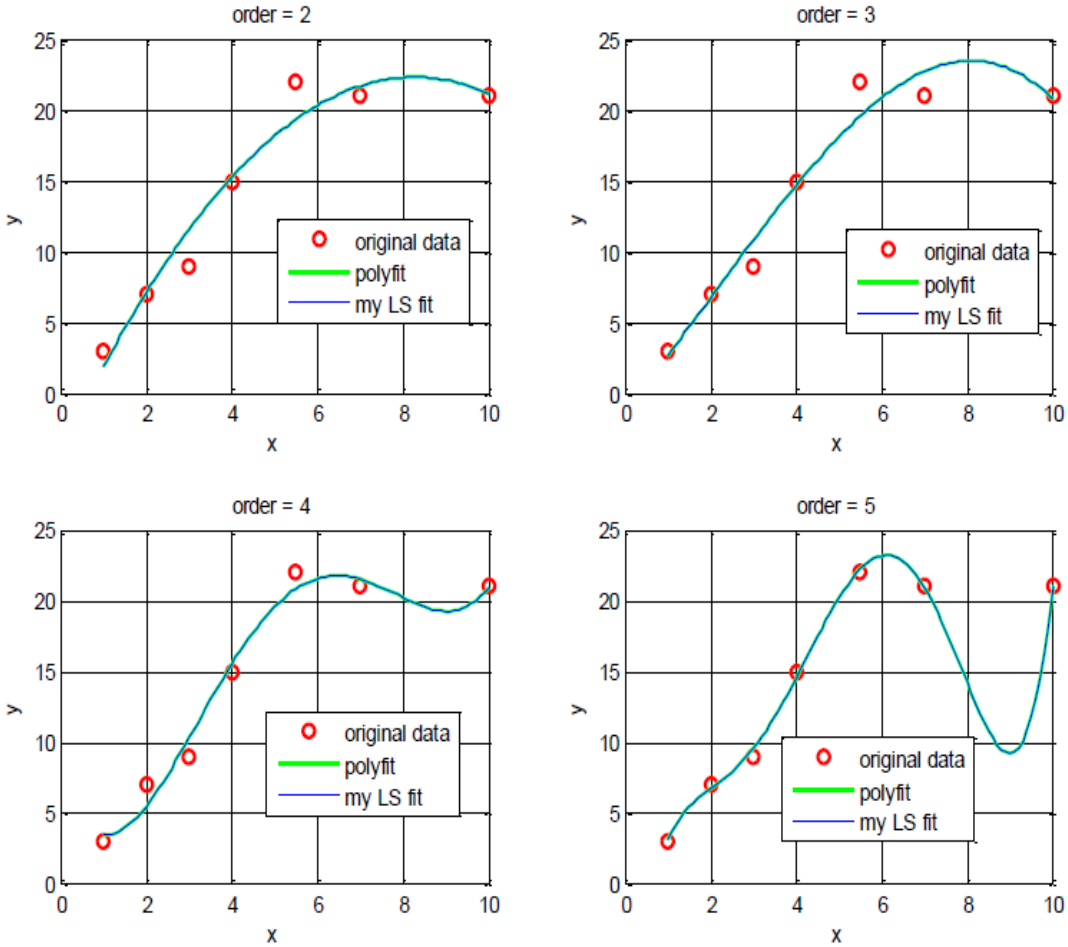
```
% see interpolated values of fits
xc = min(x): .1 : max(x);
% plot estimated polynomials
matlab_y_est = polyval(matlab_coeff, xc);
plot(xc, matlab_y_est, 'g', 'linewidth', 2) ;
my_y_est = polyval(my_coeff, xc);
plot(xc, my_y_est, 'b') ;
```

לצורך ההשוואה באופן מתמטי, חישבתי את שגיאת MSE בין הפולינומים שנמצאו בעזרת שתי הפונקציות–

```
% finding MSE
err = (my_y_est - matlab_y_est).^2 ;
mse(i) = sum(err)/length(matlab_y_est);
```

תוצאות

בחנתי את השערוכים עבור דרגות פולינום הבאות – 2,3,4,5 (1024 נקדות)
עבור כל דרגה, שערכנו פולינום בעזרת פונקציית LS שכתבנו ובעזרת פונקציית polyfit.
אנו מציגים את התוצאות בתחום הנתון יחד עם המידע המקורי–



ראשית, ניתן לראות כי שתי הפונקציות נותנות מודלים זהים בצורה כמעט מושלמת כך שהגרפים התלכדו אחד על השני.
בנוסף, וקטור השגיאות שקיבלנו הוא–

1	2	3	4
5.8934e-26	9.5964e-24	6.4081e-17	2.9409e-08

השגיאות כמעט אפסיות ולכן ניתן להסיק שהשיטה שבה השתמשנו תקינה ועובדת כראוי.

באיור המצורף קל לראות כי ככל שסדר המודל גדל, כך הוא עובר דרך יותר נקודות נתונות ובעצם השערוך יותר מדויק - בטבלה רואים שהשגיאה קטנה בהתאם.
הסיבה לכך היא שככל שסדר המודל גדל אז דרגת הפולינום גדלה אז מספר הפרמטרים (המקדמים) לייצוג המודל גדל, וכך גדל כושר התמרון של הפולינום.
למשל, הפולינום מסדר 5 עובר דרך מספר נקודות כפול מאשר פולינומים מסדר 2 ו4.

שיטות שערוך מסנן

הסבר תהליך הניסוי

בחלק זה אבציע שערוך למקדמי מסנן IIR מסוג , all-pole כלומר מסנן בעל פונקציית תמסורת מהצורה הבאה–

$$H(z) = \frac{X(z)}{E(z)} = \frac{1}{\sum_{p=1}^{P} a_p z^{-p}} = \frac{z^P}{\sum_{p=1}^{P} a_p z^{P-p}}$$

(יש רק קטבים, מלבד P אפסים באפס.)

מודל מסנן כזה נקרא model autoregressive או מודל , AR ולבעיית מציאת המקדמים קוראים בעייתAR .

המסנן המקורי שאשערך הוא מסדר 4, וחמשת המקדמים שלו הינם–

```
% original filter coefficients
A = [1 -2.7607 3.8106 -2.6535 0.9238];
```

כלומר, פונקציית התמסורת המקורית (במישור Z) היא–

$$H(z) = \frac{1}{z^4 - 2.7607z^3 + 3.8106z^2 - 2.6535z + 0.9238}$$

השערוך יתבצע בשיטות(Yule-Walker עם אלגוריתם) LD ושיטת prony בשני שלבים– בשלב הראשון סדר המסנן המשוערך ישתנה ובשלב השני משתנה אורך האות בכניסה.

בשני השלבים אנו אבציע גנרציה של רעש לבן גאוסי wn באורך מסוים–

```
% white gaussian noise
v0 = 0.2; % noise variance
wn = sqrt(v0)*randn(samples,1);
```

בנוסף, אשתמש במקדמים המקוריים כדי להציג את התגובה לתדר של המסנן המקורי–

```
% frequency response (Original)
[h,w] = freqz(1,A,samples);
plot(w/pi,20*log10(abs(h)),'k'); hold on;
ax = gca;
ax.YLim = [-100 20];
ax.XTick = 0:.5:2;
xlabel('Normalized Frequency (\times\pi rad/sample)');
ylabel('Magnitude (dB)');
```

אכניס את הרעש הלבן למסנן המקורי כדי לקבל את המוצא y—

```
% filter with H(z)=1/A(z) , wn = input and y = output
y = filter(1,A,wn);
```

כעת אשתמש במוצא y כדי לשערך את מקדמי המסנן ולהציג את התגובה לתדר,
תחילה בשיטת **YW**—

```
% AR model estimation - YW equations with LD algorithm
ARcoeffs_YW = aryule(y,order) ;

% frequency response (AR model)
[h,w] = freqz(1,ARcoeffs_YW,samples);
plot(w/pi,20*log10(abs(h)),'r'); hold on;
```

הפונקציה aryule הינה פונקציה מובנית של מטלב שתיאורה הוא–

aryule

Autoregressive all-pole model parameters — Yule-Walker method

a = aryule (x, p) returns the normalized autoregressive (AR) parameters corresponding to a model of order p for the input array, x.

הפונקציה משתמשת באלגוריתם LD כמו שמופיע בעמוד ה help -שלה–

Algorithms

`aryule` uses the Levinson-Durbin recursion on the biased estimate of the sample autocorrelation sequence to compute the parameters.

תהליך השערוך בשיטת **prony** מוצג בקטע הבא–

```
% AR model estimation - prony's method
[~,Den] = prony(y,0,order+1); % num_order = 0

[h,w] = freqz(1,Den,samples);
plot(w/pi,20*log10(abs(h)),'b');
```

הפונקציה prony הינה פונקציה מובנית של מטלב שתיאורה הוא–

prony

Prony method for filter design

[Num,Den] = prony(impulse_resp,num_ord,denom_ord) returns the numerator Num and denominator Den coefficients for a causal rational system function with impulse response impulse_resp. The system function has numerator order num_ord and denominator order denom_ord. The lengths of Num and Den are num_ord+1 and denom_ord+1.

תוצאות סדר משתנה

בשלב הראשון בחנתי את שתי השיטות עם סדרי המסנן הבאים – 2,3,4,5 ,

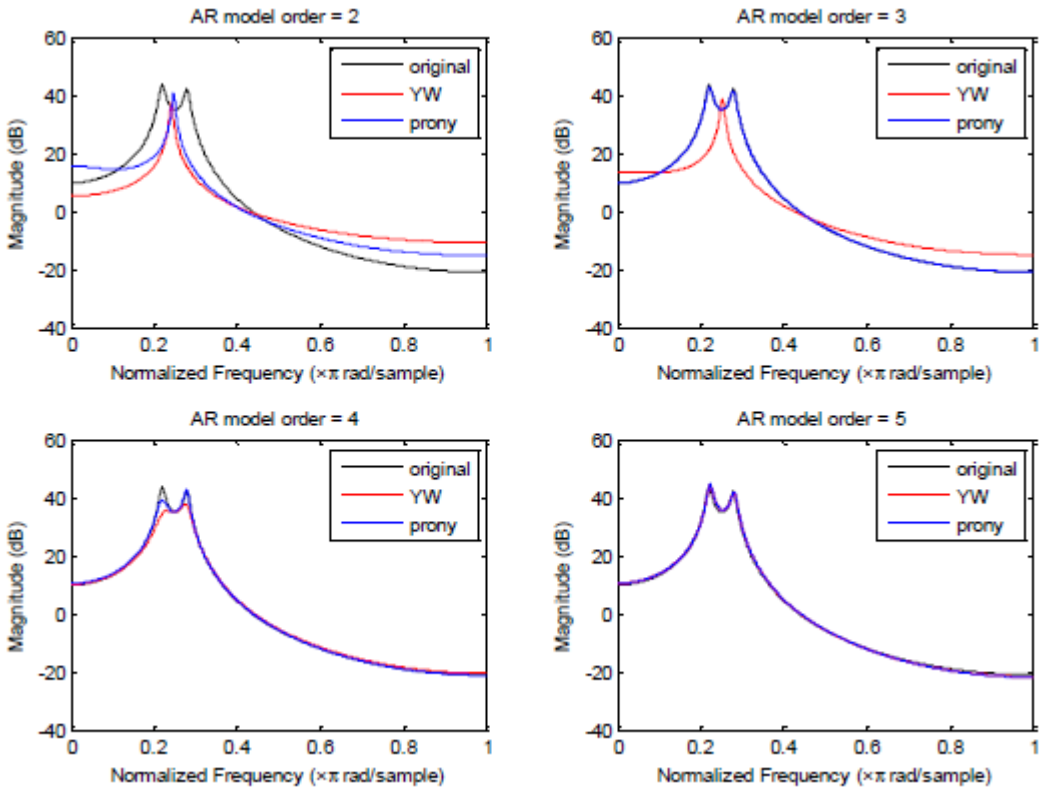
חשוב לציין שסדר המסנן המקורי שאותו שערכתי הוא 4 ואינו משתנה.

בשלב זה אורך הרעש הלבן שמוכנס למערכת המקורית הוא קבוע.

עבור כל סדר , שערכתי את המקדמים המתאימים למסנן בכל שיטה והצגתי את

תגובות התדר המתקבלות לצד תגובת המסנן המקורי.

התוצאות מוצגות באיור הבא-



מהאיור ניתן להסיק שככל שסדר המסנן גדל כך תגובת התדר המשוערכת קרובה יותר לתגובת התדר המקורית, כלומר השערוך יותר מדויק.
הסיבה לכך היא שככל שסדר המסנן גדל -> מספר המקדמים גדל -> וכך לפונקציה של תגובת התדר יש יותר 'חופש' לייצג את המסנן שגרר את התגובה y.
מסקנה זו עלתה גם בניסוי בחלק הרשון של הניסוי (כשדיברנו על סדר המסנן), כאשר עלתה דרגת הפולינום והדיוק השתפר.
בנוסף, יצא שהגרף הכחול (שיטת prony) הוא הגרף הקרוב יותר לגרף המקורי עבור כל סדר שנבדק, והחל מסדר 3 הוא משערך את תגובת התדר המקורית באיכות טובה מאוד.
לעומת זאת, הגרף האדום (שיטת YW) מהווה שערוך טוב רק החל מסדר 4, סדר המסנן המקורי, עבור סדר זה שתי השיטות הפיקו תוצאות טובות אך עדיין שיטתprony יותר קרובה למקור. ניתן להסיק מהתוצאות ששיטת prony היא שיטה טובה יותר לשערוך מודל AR כאשר סט המידע הוא סופי.
קל לראות שעבור סדר 5 , השערוכים שקיבלתי עבור שתי השיטות כמעט זהים לתגובת התדר המקורית.

על מנת לבדוק את יעילות האלגוריתמים מבחינת זמן הוספתי מקטע קוד אשר מראה לנו את הזמן שעבר בשניות עבור יול-ווקר

```
ARcoeffs_YW = zeros(1,order);
    tic;
    ARcoeffs_YW = aryule(y,order) ;
    TTT_YW1(i) = toc;
```

עבור פרוני

```
Den = zeros(1,order);
    tic;
    [~,Den] = prony(y,0,5); % num_order = 0; denom_order = 4;
    TTT_PR1(i) = toc;
```

להלן התוצאות: יול-ווקר

1	2	3	4
0.0198	0.0022	0.0011	5.7794e-04

פרוני

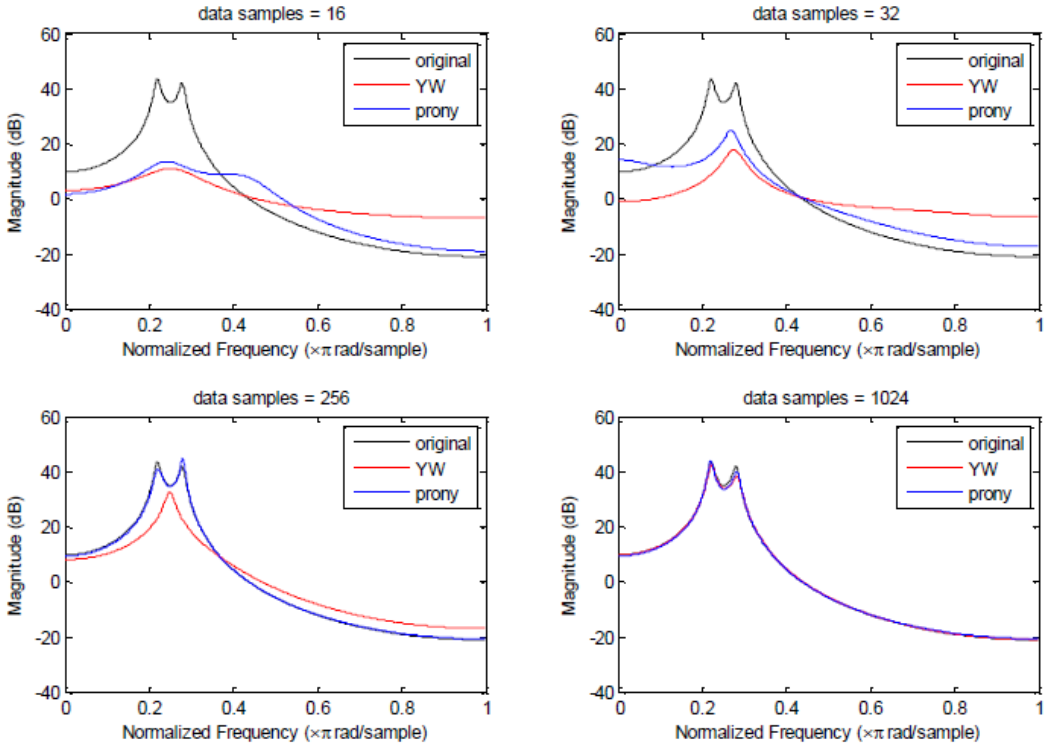
1	2	3	4
0.0178	0.0137	0.0119	0.0119

ניראה שפרוני מנצח כאשר מחפשים מקדם בודד עבור בעיית האוטו-רגרסיה, אך כפי שמצופה לפי התאוריה, ככל שסדר הבעיה גדל ניתן לראות שפרוני שומר על זמנים דומים, אך יול-ווקר אשר עושה שימוש באלגוריתם לוינסון-דרבין יורד בכמה סדרי גודל ורואים בבירור שהינו יעיל יותר מבחינת זמני חישוב.

תוצאות עם שינוי אורך אות הרעש

בחנתי את שתי השיטות כאשר השינוי הוא באורך אות הרעש הלבן שמוכנס למערכת המקורית. בשלב זה סדר המסנן בכל שערוך הוא קבוע 4.
עבור כל אורך, שערכתי את המקדמים המתאימים למסנן בכל שיטה והצגנו את תגובות התדר המתקבלות לצד תגובת המסנן המקורי.

התוצאות מוצגות באיור הבא–



מהאיור ניתן להסיק שככל שסט המידע ארוך יותר כך תגובת התדר המשוערכת קרובה יותר לתגובת התדר המקורית, כלומר השערוך מדויק יותר.
כאשר אות הכניסה מכיל 16 או 32 נקודות בלבד, שתי השיטות מתקשות לבצע שערוך למסנן עקב דלות המידע ועבור אות כניסה של 1024 נקודות קל לראות שהשערוכים שקיבלתי עבור שתי השיטות כמעט זהים לתגובת התדר המקורית.
הסיבה לכך היא שככל שסט המידע ארוך יותר, האות במבוא המערכת גורר במוצאה תגובה y בעלת יותר דגימות, כך מספקים לשיטת השערוך מידע רב יותר להיעזר בו בשערוך. ככל ששיטת השערוך מסתמכת על יותר מידע – השערוך המתקבל אמין יותר.
בנוסף, קיבלתי שהגרף הכחול(שיטתprony) הוא הגרף הקרוב יותר לגרף המקורי עבור כל סט מידע שנבדק, והחל מאורך של 256 דגימות הוא משערך את תגובת התדר המקורית באיכות טובה מאוד. לעומת זאת, הגרף האדום (שיטת YW) מהווה שערוך טוב רק כאשר אורך סט המידע הוא 1024, במצב זה שתי השיטות הפיקו תוצאות טובות אך עדיין שיטת prony יותר קרובה למקור. לכן כמו בשלב הקודם, ניתן להסיק מהתוצאות ששיטת prony היא שיטה טובה יותר לשערוך מודל AR כאשר סט המידע הוא סופי.

גם בסדרת ניסויים אלה על מנת לבדוק את יעילות האלגוריתמים מבחינת זמן הוספנו מקטע קוד אשר מראה לנו את הזמן שעבר בשניות עבור יול-ווקר

```
ARcoeffs_YW = zeros(1,order);
    tic;
    ARcoeffs_YW = aryule(y,order) ;
    TTT_YW1(i) = toc;
```

עבור פרוני

```
Den = zeros(1,order);
    tic;
    [~,Den] = prony(y,0,5); % num_order = 0; denom_order = 4;
    TTT_PR1(i) = toc;
```

להלן התוצאות: יול-ווקר

1	2	3	4
4.8838e-04	3.6602e-04	3.7413e-04	4.5840e-04

פרוני

1	2	3	4
3.8717e-04	1.8407e-04	5.7794e-04	0.0121

כפי שהיה בסדרת הניסויים הקודמת, פרוני מנצח כאשר ישנם מעט נקודות דגומות, אך כפי שמצופה לפי התאוריה, ככל שמספר דגימות, ה Data -גדל ניתן לראות שזמני החישוב בשיטת פרוני גדל (אף בכמה סדרי גודל כאשר עוברים את האלף דגימות), אך יול-ווקר אשר עושה שימוש באלגוריתם לוינסון-דרבין שומר על זמנים דומים וניתן לראות שההשפעה של עומס הנקודות זניחה ביותר, ואף זניחה יותר מהשפעת סדר הבעיה.

מסקנות וסיכום

- ב**ההסבר הכלללי** הצגתי את המודל האוטו-רגרסיבי ובעיית אוטו-רגרסיה. חקרנתי את התנהגות המודל על מנת להציגו בצורות שונות ולמצוא פתרון לבעיה. מצאתי כיהבעיה ניתנת להצגה ע"י משוואות , *Yule-Walker* ובנוסף גם בשיטת *Prony*

- על מנת לפתור את הבעיה, בשתי השיטות יש לבצע היפוך מטריצה. ידוע כי מטריצת המקדמים של משוואות *Yule-Walker* הינה טופליצית, מייצגת את מטריצת אוטו-קורלציה, ולכן ניתן לבצע עליה את אלגוריתם **.** *Levinson – Durbin Recursion* אלגוריתם זה הינו זול יותר, צורך פחות זיכרון, ומהיר יותר, פחות פעולות חישוב, מאשר היפוך ישיר של מטריצה. לעומת זאת במידה והגענו למטריצת אוטו-קורלציה שאינה טופליצית, תוצאה אשר יכולה להתקבל במידה והתהליך אינו סטציונרי בזמן, ע"י שיטת *Prony* אשר מיישמת את שיערוך *LS* אנו יכולים לפתור אנליטית את המשוואות ולמצוא את המקדמים ע"י היפוך ישיר של המטריצה – Direct Inversion

.

- עם זאת שאלגוריתם *Levinson – Durbin Recursion* יעיל יותר, שיטת*Prony* מממשת היפוך ישיר של המטריצה לכן ציפיתי לשגיאה גדולה יותר בפתרון ע"י משוואות *Yule-Walker*.

- ישנה אנלוגיה ברורה בין משוואות *Yule-Walker* לשיטת **,** *Prony* ע"י אנלוגיה זאת ניתן בחלק מן המקרים לשערך את מטריצת האוטו-קורלציה ואת וקטור האוטו-קורלציה ע"י הצגת הבעיה לפי שיטת *Prony* .

- בחלק הניסוי השתמשנו בשיטת *linear least squares* להתאמת מודל פולינומי לסט מידע נתון, תהליך הנקרא. *polynomial least squares fittings* פתרון המשוואות לחישוב המקדמים נעשה בצורה אנליטית *Direct-inversion* של היפוך מטריצת האוטו-קורלציה, פתרון המתאים גם במקרים שהמטריצה אינה טופליצית סימטרית. את התוצאות בדקנו עבור סט מידע שרירותי וכמה דרגות פולינום שונות, תוך השוואה לתוצאות פונקציה מובנית של מטלב.
- מצאתי שסדר המודל משפיע באופן משמעותי על התוצאות. ככל שהסדר גדול יותר מספר הפרמטרים לייצוג המודל גדל, וכך גדל כושר התמרון של הפולינום - כלומר השערוך מדויק יותר.
- בחלק שיטות שערוך מסנן השתמשנו בשיטת *LS* לשערוך מקדמי מסנן *IIR* מסוג *all-pole, כלומר פתירת בעיית /AR*
פתרון המשוואות לחישוב המקדמים נעשה בשיטת yule-walker (שמתייחסת למקרה הטופליצי ופותרת את המשוואות באלגוריתם *LD)* ושיטת *prony* המתאימה גם במקרים שהמטריצה אינה טופליצית סימטרית. את התוצאות בדקתי עבור מסנן שרירותי – בשלב הראשון שינינו את סדר מודל השערוך ובשלב השני שינינו את אורך סט המידע הנתון.
- מצאתי שוב שסדר המודל משפיע באופן משמעותי על התוצאות. ככל שהסדר גדול יותר – מספר הפרמטרים לייצוג המודל גדל, וכך תגובות התדר המשוערכות היו קרובות יותר למקור - כלומר השערוך מדויק יותר.
- מצאתי שאורך הסדרה משפיע באופן משמעותי על התוצאות, ככל שאורך הסדרה גדול יותר – יש כמות גדולה יותר של מידע שבעזרתו משערכים, כך השערוך מדויק יותר ותגובות התדר המשוערכות היו קרובות יותר למקור.
- בנוסף, בשני השלבים של חלק זה הגעתי למסקנה שכאשר סט המידע הוא סופי, שיטת *prony* היא שיטה טובה יותר משיטת *YW* לשערוך מודל*AR*

קוד

```
%% MATLAB PROJECT
%% yazid bisharat 315914689
%% PART 1
%%
clear all; clc; close all;
% data coordinates
x = [1 2 3 4 5.5 7 10];
y = [3 7 9 15 22 21 21];
i=1;
for order = [2 3 4 5]

     subplot(2,2,i);
     % plot given data in red
     plot(x, y, 'ro', 'linewidth', 2);grid on ;
     hold on ;
     % calculate coefficients
     matlab_coeff = polyfit(x, y, order);
     my_coeff = My_LsPolyfit(x, y, order);
     % see interpolated values of fits
     xc = min(x): .1 : max(x);
     % plot estimated polynomials
     matlab_y_est = polyval(matlab_coeff, xc);
     plot(xc, matlab_y_est, 'g', 'linewidth', 2) ;
     my_y_est = polyval(my_coeff, xc);
     plot(xc, my_y_est, 'b') ;
     % finding MSE
     err = (my_y_est - matlab_y_est).^2 ;
     mse(i) = sum(err)/length(matlab_y_est);i=i+1;
     xlabel('x'); ylabel('y');
     str = 23print('order = %d ', order);
     title(str);
     legend('original data', 'polyfit', 'my LS fit');
end
%% PART 2
clear all; clc; close all;
% original filter coefficients
A = [1 -2.7607 3.8106 -2.6535 0.9238];
%--------------changing order----------------------------
-
samples = 1024;
figure(1)
i=1;

for order = [2 3 4 5]

     % white gaussian noise
     v0 = 0.2; % noise variance
     wn = sqrt(v0)*randn(samples,1);
     subplot(2,2,i);
     % frequency response (Original)
```

```
    [h,w] = freqz(1,A,samples);
    plot(w/pi,20*log10(abs(h)),'k'); hold on;
    ax = gca;
    ax.YLim = [-100 20];
    ax.XTick = 0:.5:2;
    xlabel('Normalized Frequency (\times\pi
    rad/sample)');
    ylabel('Magnitude (dB)');
    % filter with H(z)=1/A(z) , wn = input and y =
    output
    y = filter(1,A,wn);
%----------
    % AR model estimation - YW equations with LD
    algorithm
    ARcoeffs_YW = aryule(y,order) ;
    % frequency response (AR model)
    [h,w] = freqz(1,ARcoeffs_YW,samples);
    plot(w/pi,20*log10(abs(h)),'r'); hold on;
%----------
    % AR model estimation - prony's method
    [~,Den] = prony(y,0,order+1); % num_order = 0
    [h,w] = freqz(1,Den,samples);
    plot(w/pi,20*log10(abs(h)),'b');
%----------
    str = sprintf('AR model order = %d ',order);
    title(str);
    legend('original','YW','prony');
    i = i+1;
end

%--------------changing samples-------------------------
---
figure(2)
order = 4;
i=1;
for samples = [16 32 256 1024]

    % white gaussian noise
    v0 = 0.2; % noise variance
    wn = sqrt(v0)*randn(samples,1);
    subplot(2,2,i);
    % frequency response (Original)
    [h,w] = freqz(1,A,1024);

    plot(w/pi,20*log10(abs(h)),'k'); hold on;
    ax = gca;
    ax.YLim = [-100 20];
    ax.XTick = 0:.5:2;
    xlabel('Normalized Frequency (\times\pi
    rad/sample)');
    ylabel('Magnitude (dB)');
```

```
    % filter with H(z)=1/A(z) , wn = input and y =
    output
    y = filter(1,A,wn);
%----------
    % AR model estimation - YW equations with LD
    algorithm
    ARcoeffs_YW = aryule(y,order) ;
    % frequency response (AR model)
    [h,w] = freqz(1,ARcoeffs_YW,1024);
    plot(w/pi,20*log10(abs(h)),'r'); hold on;
%----------
    % AR model estimation - prony's method
    [~,Den] = prony(y,0,order+1); % num_order = 0;
    denom_order = 4;
    [h,w] = freqz(1,Den,1024);
    plot(w/pi,20*log10(abs(h)),'b'); hold on;
    str = sprintf('data samples = %d ',samples);
    title(str);
    legend('original','YW','prony');
    i = i+1;
end
```

פונקציות:

```
function [b] = My_LsPolyfit(x,y,m)
% MyLsPolyfit(x, y, m) fits a least-squares polynomial of
degree m .
% The output is the row vector b of coefficients
[b1;b2;..;bm+1] .
% b is in descending powers Pm(x) = b1*x^m + b2*x^m-1
+...+ bm+1 .
if (length(x) ~= length(y) )
error('invlaid input data');
end
n = length(x);
y = y' ; % row vector to n-by-1 vector of responses.
% X is the n-by-m+1 design matrix for the model

X = zeros(n,m+1);
for i = 1:n
    for j = 1:m+1
        X(i,j) = x(i)^(m+1-j);
    end
end
if (det(X'*X) == 0 )
error('there is no inverse matrix for solution');
end
% y = X*b -> X'*y = X'X*b -> b = inv(X'X*)*X'*y
b = fliplr( inv(X'*X)*X'*y );
en
```

References

[1] "https://www.ece.iastate.edu/~namrata/EE524/index.html".

[2] http://www.iste.co.uk/data/doc_ukqmvpysztlg.pdf, "http://www.iste.co.uk/data/doc_ukqmvpysztlg.pdf".