

Department of
Mathematics



היחידה
למתמטיקה

**פרויקט מסכם לתואר בוגר במדעים (B.Sc)
במתמטיקה שימושית**

**שימוש בגוזר מצב החלקה בחוקי הנחיית טילים
אופיר חזן**

**Using the sliding mode differentiator
technique in missile guidance laws**

Ofir Hazan

פרויקט מסכם לתואר בוגר במדעים (B.Sc)

במתמטיקה שימושית

שימוש בגוזר מצב החלקה בחוקי הנחיית טילים

אופיר חזן

**Using the sliding mode differentiator
technique in missile guidance laws**

Ofir Hazan

Advisor:

Prof. Vladimir Turetsky

מנחה:

פרופ'ח ולדימיר טורצקי

Karmiel

2012

כרמיאל

תוכן עניינים

1.	מבוא	5
2.	הגדרת הבעיה	6
2.1	מתאר סכמטי עבור הבעיה במישור $x - y$	6
2.2	המרחק היחסי בין הגופים y	7
2.3	מערכת ההתנגשות	7
2.4	קבועי המערכת ופונקציית הבקרה v	9
3.	חוק ההנחיה של המיירט U	10
3.1	חוק ההנחיה במערכת זו	10
3.2	משתני מצב לא ידועים בחוק ההנחיה U	10
4.	מנגנון שחזור הנגזרות - The Sliding Mode Differentiator	11
4.1	רקע לשיטה ותרשים מייצג	11
4.2	מערכת גוזר ההחלקה	12
4.3	דוגמאות לשימוש בגוזר ההחלקה	13
4.3.1	דוגמא ראשונה	13
4.3.2	דוגמא שנייה	14
4.3.3	דוגמא שלישית	15
4.4	קוד תוכנית מממשת – Matlab – Euler_Sliding_Mode	17
4.5	מסקנות ביניים ושאלות מנחות	19
4.6	השתלבות הגוזר במערכת הכוללת	19
5.	קבועי הגוזר Differentiation parameters	20
5.1	קוד תוכנית מממשת – Matlab – Levant_lamda_criterion	21
5.2	מטרת התוכנית Levant_lamda_criterion ודרך פעולתה	24
6.	הרחבת המודל עבור λ ות משתנות בזמן	25
6.1	מוטיבציה ליצירת מנגנון "הסתגלות"	25
6.2	תבנית $\lambda i(t)$ נבחרות עבור המערכת PE	25
6.3	בחינת ההשערה על מערכת PE	25
6.3.1	דרך המבחן	25
6.3.2	מימוש המבחן על המערכת PE, דוגמא	26
6.3.3	טבלת תוצאות השוואתית בין שתי ההרצות	26

27.....	6.3.4 השוואת גרפי התפלגות המצטברת בין שתי ההרצות
27.....	6.3.5 מסקנות המבחן ורעיונות להמשך
28.....	7. הדמיות שיגור
28.....	7.1 מספר דוגמאות הדמייה תחת תנאים שונים
29.....	7.2 קוד תוכנית מממשת – Matlab – pursuer_evader_movie
34.....	8. סיכום
35.....	9. ביבליוגרפיה



1. מבוא

בקרה בתנאי אי ודאות הינה אחד הנושאים העיקריים של תורת הבקרה המודרנית. בפרויקט זה אציג את בעיית הרודף נרדף (טיל מיירט - Pursuer וטיל מיורט - Evader ובקצרה PE) בעיה זו מבוססת על מודל בו שני גופים נעים אחד לעבר השני כאשר מטרת הגוף המיירט היא לייצר פגיעה בגוף המיורט טרם הגעתו לקרקע או טרם חליפתם אחד על פני השני. מערכת זו הינה מערכת בקרה דינאמית לא ליניארית. המודל הנידון מתבצע תחת כמה הנחות בסיסיות :

- המצאות במישור הדו מימדי.
- מיקום הגופים ידוע בכל שלב תעופתם.
- מהירויות הטילים קבועות ותאוצתם חסומות בגודלן.
- מסלולי התעופה של שני הטילים ניתנים ללניאריזציה ביחס לגיאומטריית מסלולם.

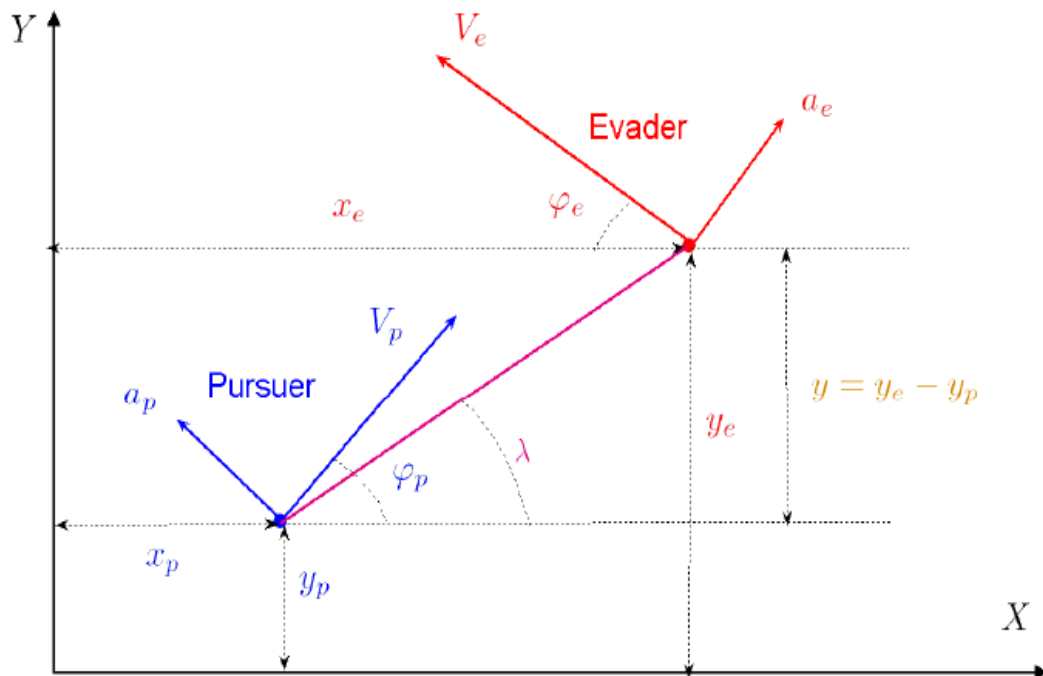
הערה : תמרון הגופים מתבצע באמצעות תאוצות רדיאליות קטנות. אין אפשרות לתמרון לאחר (אלא רק כלפי מעלה או מטה) הן עבור הרודף והן עבור הנרדף.

העבודה בתכליתה, תציג את השימוש במנגנון שחזור הנגזרות, Sliding Mode Differentiator, כפי שהוצג במאמרו של A. Levant, ככלי מרכזי לשחזור נגזרות של אות קלט בעל תכונות לא ידועות ושילוב הגוזר במערכת הכוללת. מנגנון זה, תוך בחירת הפרמטרים המתאימים יביא לתוצאת יירוט של הטיל הרודף בטיל הנרדף. במקביל יוצגו מימושים באמצעות תוכנה והצגת הדמיות (סימולציות) המשלבות שינויים בקבועי הגוזר (ואף הרחבתם לפונקציות המשתנות בזמן) ובפרמטרים נוספים.

2. הגדרת הבעיה

הבעיה הניצבת לפנינו הינה משחק שיגור, המשחק מתבצע במישור הדו ממדי. את המשתתפים נייצג כשני חלקיקים (גופים חסרי ממדים) אשר ידמו טילים. טיל אחד רודף (Pursuer) והשני נרדף (Evader), אשר משוגרים זה אל מול זה. מטרת הטיל הרודף היא יירוט מדויק במטרה, בכפיפות לפונקצית בקרה עצמית (שהיא תוצר של משחקים דיפרנציאליים) אשר ידועה לנו. תוך אי ידיעת מרכיבי תעופת האויב פרט להנחת חסימות פונקציית בקרתו שהיא פונקציה שרירותית.

2.1 מתאר סכמטי עבור הבעיה במישור $x - y$



מציינים את מיקומם של הרודף והנרדף בהתאמה. $(x_e, y_e), (x_p, y_p)$

הן מהירויות הגופים בהתאמה. v_e, v_p

הן התאוצות הצידיות של הרודף והנרדף בהתאמה. a_e, a_p

הן הזוויות הצידיות של הרודף והנרדף ביחס לקו הראיה. φ_e, φ_p

מתאר את ההפרדה היחסית בין הטילים. $y = y_e - y_p$

2.2 המרחק היחסי בין הגופים y

את המרחק בין הגופים כתלות במרחק r וזוויות הראייה האופקית λ ניתן לתאר באמצעות שיקולי גיאומטריה פשוטים תחת ההנחה כי בשל המרחק הגדול בין הטילים מדובר בזוויות קו ראייה קטנות, כך שמתקיים:

$$\lambda \cong \tan(\lambda) = \frac{y_e - y_p}{x_e - x_p} \cong \sin(\lambda) = \frac{y}{r} \rightarrow y \cong r \cdot \lambda$$

הערך הנמדד הוא זווית קו הראייה λ , ערך זה מורעש משום שבמציאות בשל שגיאות מדידה נקבל ערך מקורב. לאחר הוספת רעש אקראי למדידת זווית קו הראייה האופקית והסימונים מתקבל:

ρ רעש אקראי (בסימולציות נעשה שימוש ברעש בעל התפלגות אחידה)

σ קבוע המייצג את גודל הרעש, בסימולציות נבחר כעשירית מילי רדיאן

$$\lambda^* = \lambda + \sigma \cdot \rho$$

$$y^* \cong r \cdot \lambda^*$$

כפי שצוין, הבעיה היא בעיית יירוט. ברצוננו להנחות את המיירט כך שבזמן סופי t_f נקבל התנגשות כלומר $y^*(t_f) = 0$.

2.3 מערכת ההתנגשות

את מערכת ההתנגשות (s), ניתן לתאר באמצעות מערכת משוואות דיפרנציאליות הבאה:

(V.Turetsky, J.Shinar 2003)

מעבר ממערכת (s) למערכת (*) נעשה ע"י לניאריזציה. תוך שימוש במטריצת אקספוננט. (פיתוח זה אינו חלק מפרויקט זה, להרחבה ראה פרויקט סוף: "פיתוח אסימפטוטי לפתרון למשוואה דיפרנציאלית של מודל רודף נרדף בסביבת נקודה סינגולארית" מאת חיים כהן)

$$\left\{ \begin{array}{lll} \frac{dx_p}{dt} = & V_p \cos \varphi_p, & x_p(0) = 0, \\ \frac{dy_p}{dt} = & V_p \sin \varphi_p, & y_p(0) = 0, \\ \frac{d\varphi_p}{dt} = & a_p / V_p, & \varphi_p(0) = \varphi_{p0}, \\ \frac{da_p}{dt} = & (a_p^{\max} u - a_p) / \tau_p, & a_p(0) = 0, \\ \frac{dx_e}{dt} = & -V_e \cos \varphi_e, & x_e(0) = r_0, \\ \frac{dy_e}{dt} = & V_e \sin \varphi_e, & y_e(0) = 0, \\ \frac{d\varphi_e}{dt} = & a_e / V_e, & \varphi_e(0) = \varphi_{e0}, \\ \frac{da_e}{dt} = & (a_e^{\max} v - a_e) / \tau_e, & a_e(0) = 0 \end{array} \right. \quad (s)$$

הפונקציות $u(t)$ ו- $v(t)$ הן הבקורות של הרודף והנרדף בהתאמה, עבורן מתקיים:

$$|u(t)| \leq 1, \quad 0 \leq t \leq t_f$$

$$|v(t)| \leq 1, \quad 0 \leq t \leq t_f$$

כלומר, פונקציות הבקרה הן מדידות וחסומות בטווח הזמן $[0, t_f]$.

(*)

$$\begin{array}{ll} \dot{x}_1 = x_2, & x_1(0) = 0 \\ \dot{x}_2 = x_3 - x_4, & x_2(0) = V_e \sin \phi_e^0 - V_p \sin \phi_p^0 \\ \dot{x}_3 = (a_e^c - x_3) / \tau_e, & x_3(0) = 0 \\ \dot{x}_4 = (a_p^c - x_4) / \tau_p, & x_4(0) = 0 \end{array}$$

2.4 קבועי המערכת ופונקציית הבקרה v

הקבועים שנבחרו לאבחון הבעיה כנתונים מוצגים בטבלה הבאה :

מרחקים	מהירויות	תאוצות וקבועי זמן	זוויות התחלתיות
$r_0 = 20[\text{km}] = 20000[\text{m}]$	$v_p = 2300[\frac{\text{m}}{\text{s}}]$	$a_p^{\max} = 200[\frac{\text{m}}{\text{s}^2}]$	$\varphi_p(0) = 0$
$x_p(0) = 0[\text{m}]$	$v_e = 2700[\frac{\text{m}}{\text{s}}]$	$a_e^{\max} = 100[\frac{\text{m}}{\text{s}^2}]$	$\varphi_e(0) = 0$
$y_p(0) = 0[\text{m}]$		$a_p(0) = 0[\frac{\text{m}}{\text{s}^2}]$	
$x_e(0) = r_0[\text{m}]$		$a_e(0) = 0[\frac{\text{m}}{\text{s}^2}]$	
$y_e(0) = 0[\text{m}]$		$\tau_p = 0.2[\frac{\text{m}}{\text{s}^2}]$	
		$\tau_e = 0.2[\frac{\text{m}}{\text{s}^2}]$	

3. חוק ההנחיה של המיירט U

3.1 חוק ההנחיה במערכת זו

קיימים כמה חוקי הנחיה (בקרה) U של המיירט אשר מבטיחים פגיעה בזמן הסופי t_f . תאורתית, ובהינתן מידע מלא על משתני המצב y, v, a ידוע כי חוק ההנחיה כפי שיוצג בבעיה זו פותר את המודל ומביא ליירוט מדויק. בפרויקט זה נבחר חוק ההנחיה שהינו תוצר של משחקים דיפרנציאליים, מהצורה:

$$U = \text{sign}(Z)$$

כאשר Z נגזר ממטריצה פונדמנטאלית (מטריצת קושי):

$$Z = y + (t_f - t)y' + \tau_e^2 \left[e^{-\frac{(t_f-t)}{\tau_e}} + \frac{(t_f-t)}{\tau_e} - 1 \right] a_e - \tau_p^2 \left[e^{-\frac{(t_f-t)}{\tau_p}} + \frac{(t_f-t)}{\tau_p} - 1 \right] a_p$$

$$Z = Z(y^*, y'^*, a_e^*, a_p)$$

$$\text{sign}(Z) = \begin{cases} 1, & Z > 1 \\ Z, & -1 \leq Z \leq 1 \\ -1, & Z < -1 \end{cases}$$

קל לראות שהביטוי ל Z מכיל משתני מצב שמידע עליהם אינו קיים ולכן עלינו לספק מנגנון לשחזור מידע זה בזמן אמת (בעת תעופה).

מידע על משתני המצב צריך להיות זמין בכל זמן השיגור $t \in [0, t_f]$

כלומר, בכדי לקבל תפקוד נכון של חוק ההנחיה U יש צורך בידיעת מהירותו ותאוצתו של הנרדף ולכן עלינו לשחזר בקירוב טוב את y', y'', a_e ומהם להרכיב את y', a_e

3.2 משתני מצב לא ידועים בחוק ההנחיה U

כיצד נקבל את משתני המצב הדרושים?

נשים לב כי התאוצה העצמית a_p ידועה לנו, המשתנים t, t_f, τ_e, τ_p ידועים גם כן ומהנחות

המודל ידוע גם המיקום y המקיים $y = y_e - y_p$. לכן כל שנותר הוא לקבל את y', a_e

את הצורך בקבלת נגזרת הראשונה ניתן לראות בבירור בביטוי $(t_f - t)y'$ ואילו צורך בידיעת

הנגזרת השנייה ישנו בביטוי לתאוצת הנרדף a_e כמתואר:

$$y = y_e - y_p$$

$$y' = y'_e - y'_p = v_e \sin(\varphi_e) - v_p \sin(\varphi_p)$$

$$y'' = y''_e - y''_p \rightarrow y'' = a_e - a_p \rightarrow a_e = y'' + a_p$$

4. מנגנון שחזור הנגזרות - The Sliding Mode Differentiator

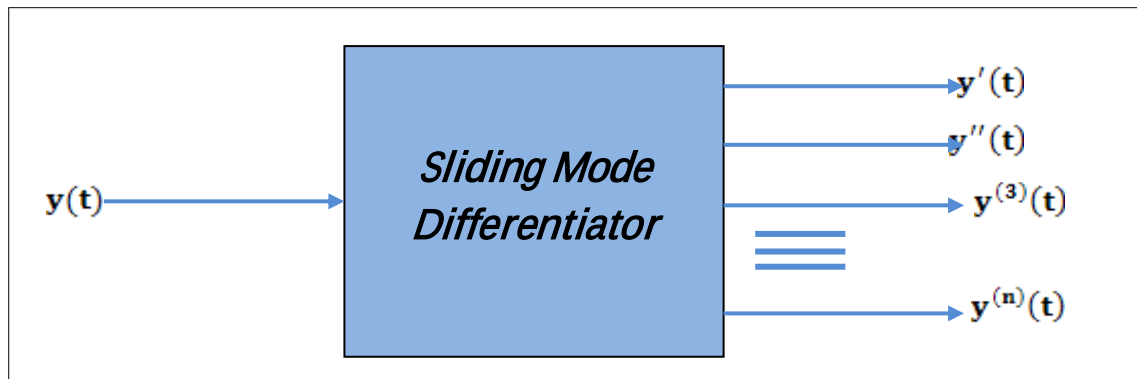
4.1 רקע לשיטה ותרשים מייצג

גישת שחזור הנגזרות של אות קלט באמצעות גוזר ההחלקה הוצגה בתחילה על ידי Levantovsky בשנת 1985 ועל ידי Levant החל משנת 1993 תוך הרחבות שבוצעו בה מאותה שנה ועד לשנים האחרונות בעבודתם הפורה של Levant ושותפיו למחקר.

גישה זו הינה גישה לשחזור מידע עבור מערכות לא ליניאריות מורכבות, שאינן פרמטריות המגלמות בתוכן אי וודאיות לגבי הדינאמיקה של המערכת, הגישה הוכיחה עצמה כמועדפת על פני שיטות שחזור מידע אחרות רבות ולכן בפרויקט זה הוחלט להתמקד השימוש בה ובבחינת השערות על מערכת היירוט PE.

עבור אות קלט, נסמנו $y(t)$, פונקציה המוגדרת בקטע $[0, \infty]$ עם תכונות לא ידועות, המכילה רעש רנדומאלי גם כן בעל תכונות שאינן ידועות נוכל לשחזר בקירוב טוב את נגזרותיה של y . המטרה היא למצוא בזמן אמת הערכות של הנגזרות $y'(t), y''(t), y^{(3)}(t) \dots y^{(n)}(t)$ אשר יהיו נגזרות מדויקות בהעדר רעשי המדידה.

את שיטת הגזירה הנומרית ניתן להמחיש באמצעות התרשים הבא:



ניתן לראות כקלט אות שהינו פונקציה $y(t)$ עם תכונות לא ידועות והפלט הינו הנגזרות עד לסדר הרצוי, בהתאם למערכת שתוצג בסעיף הבא.

4.2 מערכת גיזר ההחלקה

מערכת זו הושגה לאחרונה עי Arie Levant בשנת 2002 והוצגה במאמרו Higher-order sliding modes שפורסם בשנת 2003 בצורה הבאה :

$$\begin{aligned} \frac{dz_0}{dt} &= w_0 \\ w_0 &= -\lambda_0 \cdot |z_0 - y(t)|^{\frac{n}{n+1}} \cdot \text{sign}(z_0 - y(t)) + z_1 \frac{dz_1}{dt} = w_1 \\ w_1 &= -\lambda_1 \cdot |z_1 - w_0|^{\frac{n-1}{n}} \cdot \text{sign}(z_1 - w_0) + z_2 \\ &\dots \\ \frac{dz_{n-1}}{dt} &= w_{n-1} \\ w_{n-1} &= -\lambda_{n-1} \cdot |z_{n-1} - w_{n-2}|^{\frac{1}{2}} \cdot \text{sign}(z_{n-1} - w_{n-2}) + z_n \\ \frac{dz_n}{dt} &= -\lambda_n \cdot \text{sign}(z_n - w_{n-1}) \end{aligned}$$

לפי משפטו של Levant קיימים קבועים (λ_i) מספיק גדולים כך שהגזר מבצע גזירה טובה (מקבל את נגזרותיה של y כדרוש).

דיון על ערכי הקבועים המודגשים באדום ייתן בהרחבה בהמשך.

במערכת מתקיים כי $z_0 = y(t)$, $z_1 = y'(t)$, ..., $z_n = y^{(n)}(t)$ זאת לאחר הורדת סדר של משוואה דיפרנציאלית מסדר n .

כמקרה פרטי של המערכת, נקבע את n להיות 2 וכך נוכל לשחזר את הנגזרות $y'(t)$, $y''(t)$. המערכת המתאימה הינה :

$$\begin{aligned} \frac{dz_0}{dt} &= w_0 \\ w_0 &= -\lambda_0 \cdot |z_0 - y(t)|^{\frac{2}{3}} \cdot \text{sign}(z_0 - y(t)) + z_1 \\ \frac{dz_1}{dt} &= w_1 \\ w_1 &= -\lambda_1 \cdot |z_1 - w_0|^{\frac{1}{2}} \cdot \text{sign}(z_1 - w_0) + z_2 \\ \frac{dz_2}{dt} &= -\lambda_2 \cdot \text{sign}(z_2 - w_1) \end{aligned}$$

4.3 דוגמאות לשימוש בגוזר ההחלקה

מדדי טיב הגזירה שנבחנו הינם: SSE (סכום שגיאות מינימאלי), פונקציית ההצטברות של מרחק ההחטאה (Cumulative Distribution Function) וממד גרפי. תזכורת SSE:

$$\sigma_1^{\text{calc}} = \sqrt{\sum (y' - Z_1)^2} \quad \text{- עבור גזרת ראשונה}$$

$$\sigma_2^{\text{calc}} = \sqrt{\sum (y'' - Z_2)^2} \quad \text{- עבור גזרת שנייה}$$

4.3.1 דוגמא ראשונה

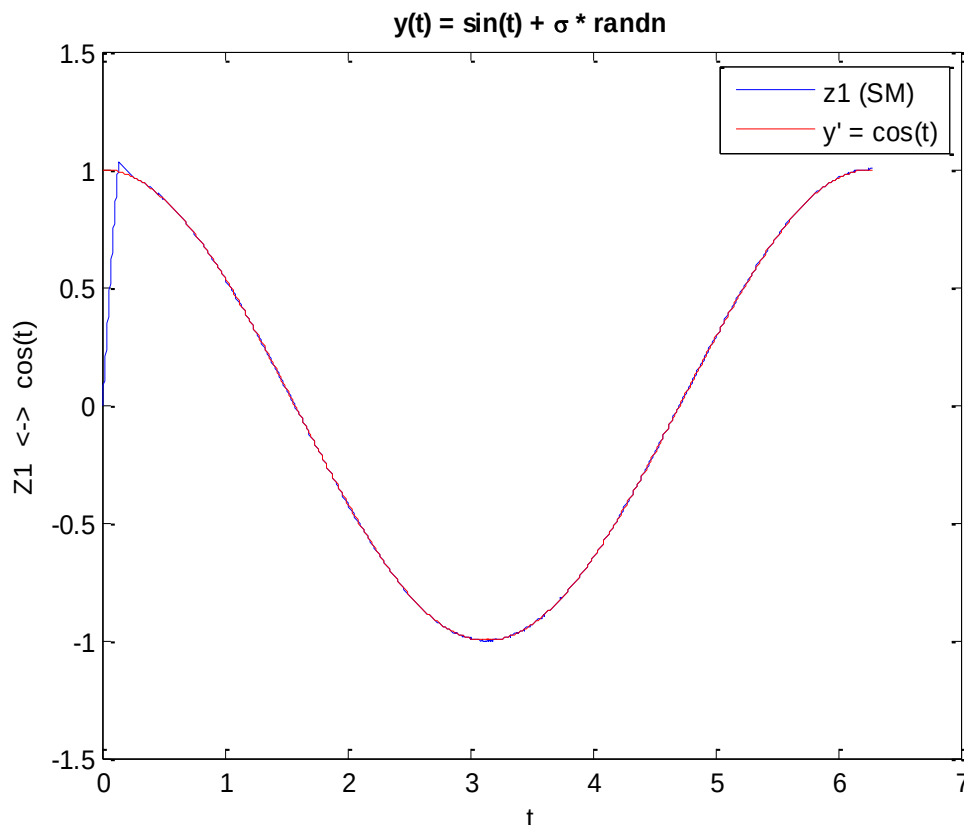
כדוגמא ראשונה נגדיר כאות קלט את $\sin$ מורעש בצורה הבאה: $y(t) = \sin(t) + \sigma \cdot \rho$

בתוספת הצבת הקבועים: $\lambda_0 = 12, \lambda_1 = 10, \lambda_2 = 10$

וגודל רעש אקראי של עשירית מילי רדיאן $\sigma = 1e^{-4}$

לאחר הצבת הקבועים במערכת ■■ ופתרון המערכת בצורה נומרית באמצעות שיטת אוילר (Euler) מסדר ראשון ב Matlab התקבלו התוצאות הבאות:

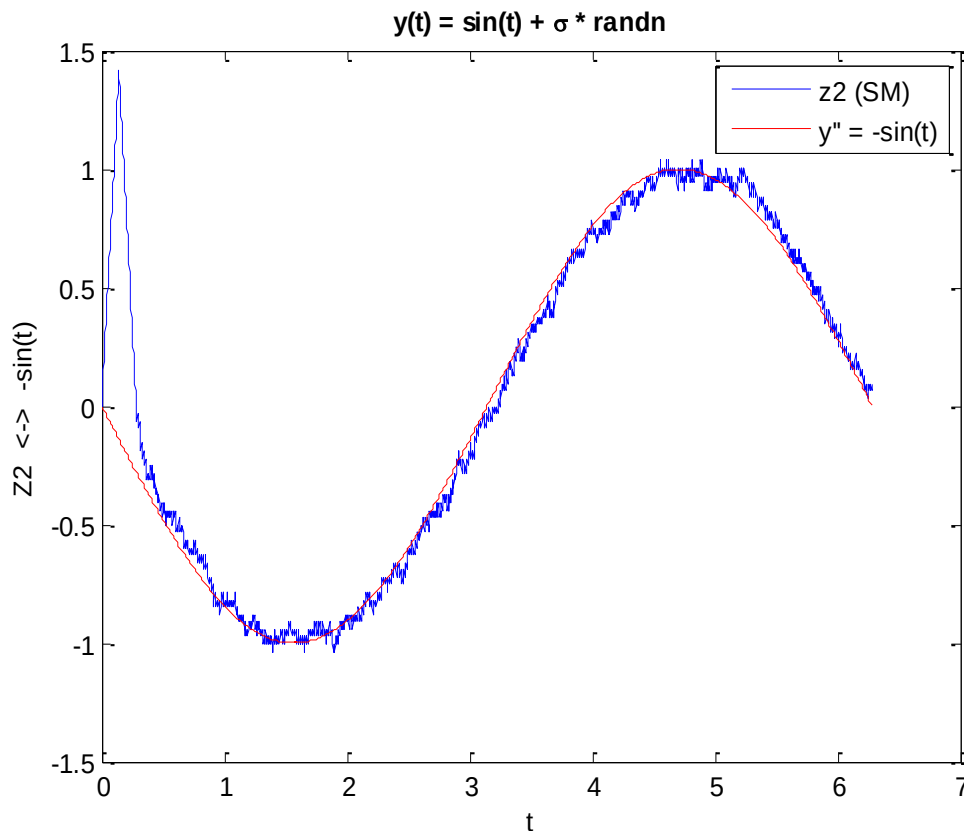
נגזרת ראשונה:



ניתן לראות החלקה כמעט מושלמת בין הנגזרת המדויקת לבין התוצר של גוזר ההחלקה בשל ההפרשים הקטנים שהתקבלו בין שני הפונקציות.

$$\sigma_1^{\text{calc}} = 0.01$$

נגזרת שנייה :



$$\sigma_2^{\text{calc}} = 0.04$$

גם במקרה זה, ההחלקה עבור הנגזרת השנייה הינה טובה. יחד עם זאת קל לראות את ביטוי של הרעש האקראי בתוצרי החישוב. כמובן שלתוצאות הגזירה יש חשיבות רבה בעוצמת הרעש האקראי וכמובן גם בקבועי הגזור, דבר שיומחש בהמשך.

4.3.2 דוגמא שנייה

כפי שהוזכר קודם לכן, לפי Levant קיימים קבועים, מספיק גדולים כך שהגזור מבצע גזירה טובה.

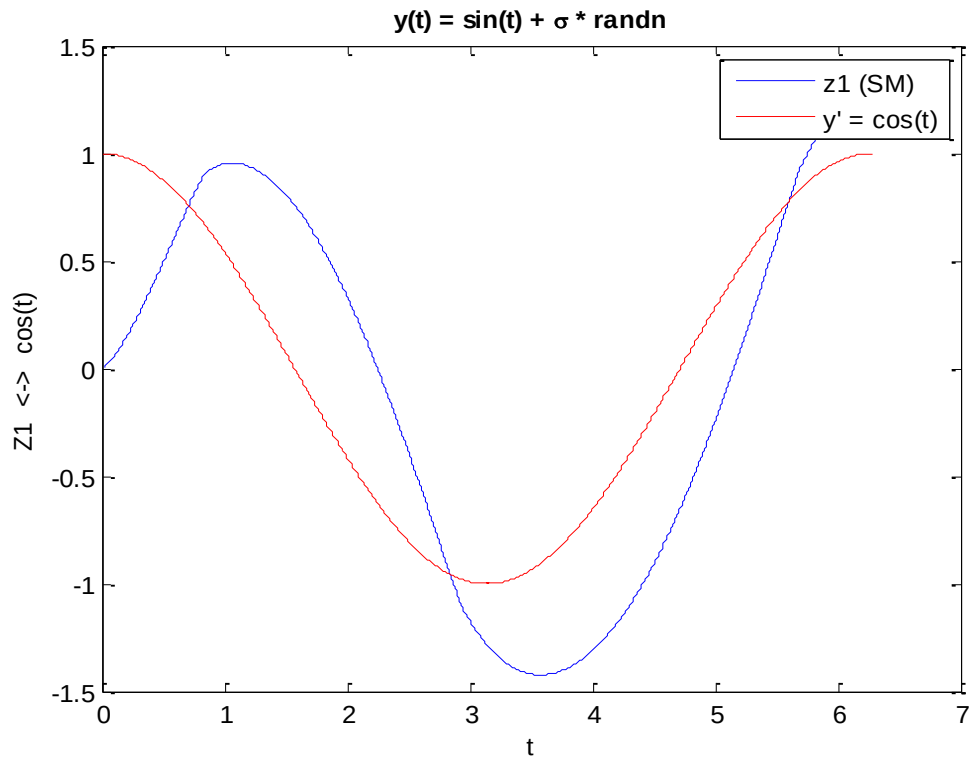
בדוגמא זו נראה מצב בו בחירת הקבועים במערכת ■■■ הינה קריטית לתוצאות הגזור.

נגדיר כאות קלט אות $\sin$ מורעש בצורה הבאה : $y(t) = \sin(t) + \sigma \cdot \rho$

כעת, נבחר את הקבועים אשר אינם מספיק גדולים לאות הנתון : $\lambda_0 = 3, \lambda_1 = 1, \lambda_2 = 1$

וגודל רעש אקראי של עשירית מילי רדיאן $\sigma = 1e^{-4}$

כעת, לאחר הצבת הקבועים במערכת ■■■ ופתרון המערכת בצורה נומרית באמצעות שיטת אוילר (Euler) מסדר ראשון ב Matlab ניתן לראות תוצאות גזירה לא טובות כבר עבור הנגזרת הראשונה! כמוצג בגרף הבא :



$$\sigma_1^{\text{calc}} = 0.28$$

$$\sigma_2^{\text{calc}} = 0.78$$

הערה : נגזרת שנייה במצב גרוע עוד יותר.

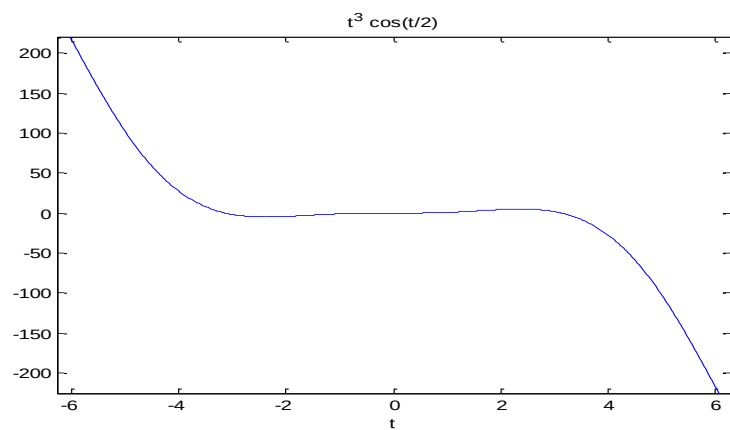
4.3.3 דוגמא שלישית

כדוגמא שלישית נגדיר כאות קלט פונקציה מהצורה : $y(t) = \cos\left(\frac{t}{2}\right) \cdot t^3 + \sigma \cdot \rho$

בתוספת הצבת הקבועים : $\lambda_0 = 30, \lambda_1 = 5, \lambda_2 = 30$

וגודל רעש אקראי של מילי רדיאן $\sigma = 1e^{-3}$

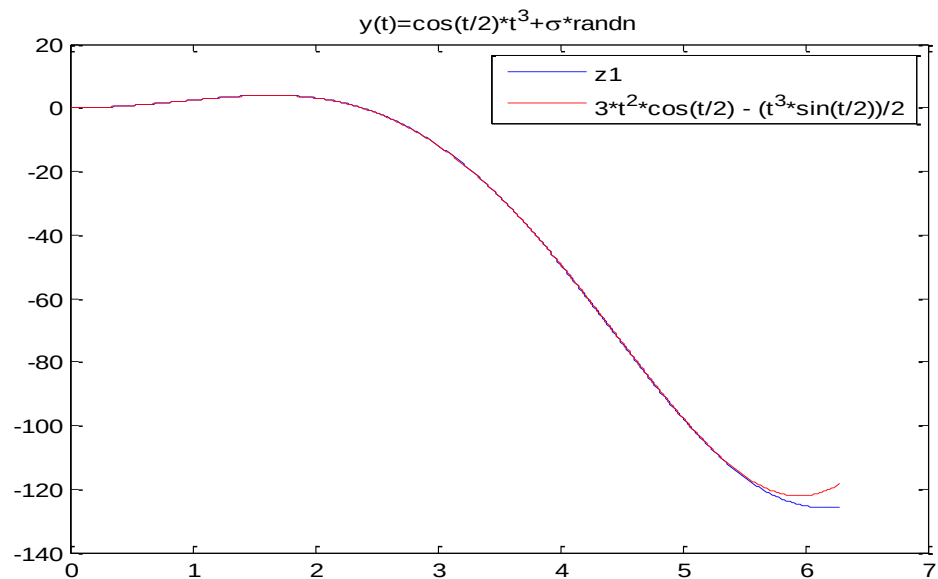
פונקציית הקלט נראת



כך

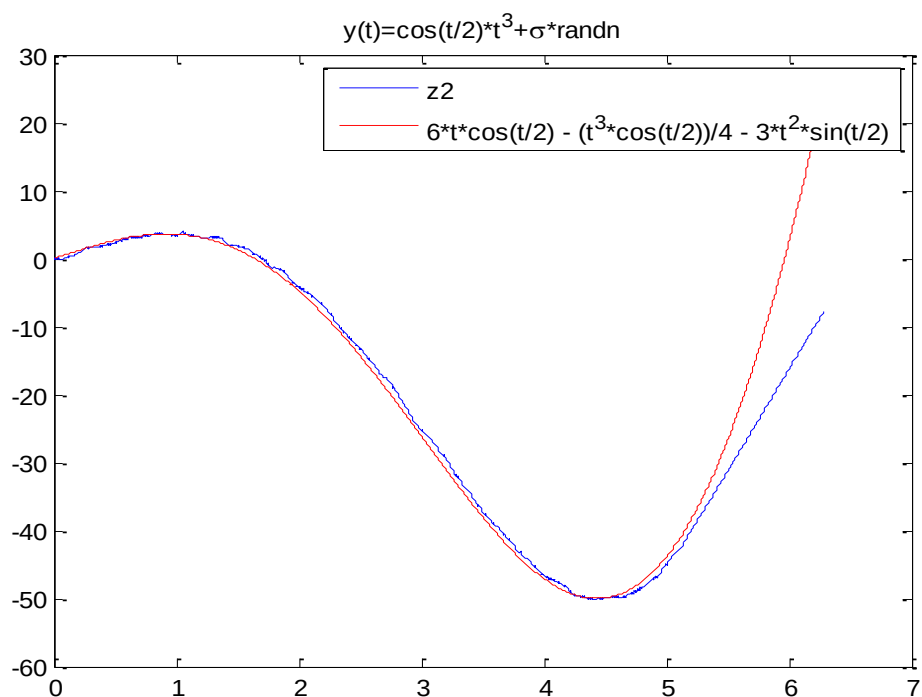
לאחר הצבת הקבועים במערכת ■ ■ ופתרון המערכת בצורה נומרית באמצעות שיטת אוילר (Euler) מסדר ראשון ב Matlab התקבלו התוצאות הבאות :

נגזרת
ראשונה :



$$\sigma_1^{\text{calc}} = 0.01$$

נגזרת
שנייה :



$$\sigma_2^{\text{calc}} = 0.03$$

Euler_Sliding_Mode – Matlab – תוכנית מממשת – קוד 4.4

```
function Euler_Sliding_Mode(choose)

% for solving Equation
% RUN: >>Euler_Sliding_Mode(choosen_number_as_integer)
% choose      1->y(t)=sin(t)
%             2->y(t)=y(t)=cos(t/2)*t^3+sigma*randn
%             3->y(t)=t^2
%             4->y(t)=sin(t)+eps*sin(1000*t)
%             5->y(t)=sin(t)+sigma*randn
%             6->y(t)=t^2+sigma*randn

clc, close all
t=0:pi/1000:2*pi;
h=(t(end)-t(1))/length(t);
z0(1)=0;
z1(1)=0;
z2(1)=0;

lam0=30;
lam1=5;
lam2=30;

sigma=1e-3;
eps=10;

% A - different inputs signals y(t)
for i=1:length(t)-1
    switch choose % Set w0
        case 1
            w0=-lam0*[abs(z0(i)-sin(t(i))))^(2/3) * sign(z0(i)-sin(t(i)))+z1(i);
        case 2
            w0=-lam0*[abs(z0(i)-(cos(t(i)/2)*t(i)^3+sigma*randn))]^(2/3) * sign(z0(i)-
            (cos(t(i)/2)*t(i)^3+sigma*randn))+z1(i);
        case 3
            w0=-lam0*[abs(z0(i)-t(i)^2)]^(2/3) * sign(z0(i)-t(i)^2)+z1(i);
        case 4
            w0=-lam0*[abs(z0(i)-(sin(t(i))+eps*sin(1000*t(i))))]^(2/3) * sign(z0(i)-
            (sin(t(i))+eps*sin(1000*t(i))))+z1(i);
        case 5 %random noise (y(t)=sin(t))
            w0=-lam0*[abs(z0(i)-(sin(t(i))+sigma*randn))]^(2/3) * sign(z0(i)-
            (sin(t(i))+sigma*randn))+z1(i);
        case 6 %random noise (y(t)=t^2)
            w0=-lam0*[abs(z0(i)-((t(i)^2)+sigma*randn(1,1)))]^(2/3) * sign(z0(i)-
            ((t(i)^2)+sigma*randn(1,1)))+z1(i);
        end
        z0(i+1)=z0(i)+h*w0;

        w1=-lam1*abs(z1(i)-w0)^(1/2)*sign(z1(i)-w0)+z2(i); %Set w1
        z1(i+1)=z1(i)+h*w1;

        w2=-lam2*sign(z2(i)-w1); %set w2
        z2(i+1)=z2(i)+h*w2;
    end

subplot(2,2,1); plot(t,z0)
title('z0=y(t)')
subplot(2,2,2); plot(t,z1)
title('z1=y'(t)')
subplot(2,2,3); plot(t,z2)
title('z2=y''(t)')
figure
plot(t,z0,t,z1,t,z2), legend('z0','z1','z2')
```

```

switch choose
    case 1
        disp('y(t)=sin(t)')
        figure
        plot(t,z1,t,cos(t)),title('y(t)=sin(t)'), legend('z1','cos(t)')
        figure
        plot(t,z2,t,-sin(t)),title('y(t)=sin(t)'), legend('z2','-sin(t)')
    case 2
        disp('y(t)=cos(t/2)*t^3+\sigma*randn')
        figure
        plot(t,z1(1:length(t)), 'b', t, 3*t.^2.*cos(t/2)- (t.^3.*sin(t/2))./2, 'r')
        title('y(t)=cos(t/2)*t^3+\sigma*randn')
        legend('z1','3*t^2*cos(t/2) - (t^3*sin(t/2))/2')
        figure
        plot(t,z2(1:length(t)), 'b', t, 6*t.*cos(t/2)-(t.^3.*cos(t/2))./4 -
3.*t.^2.*sin(t/2), 'r')
        title('y(t)=cos(t/2)*t^3+\sigma*randn')
        legend('z2','6*t*cos(t/2) - (t^3*cos(t/2))/4 - 3*t^2*sin(t/2)')
    case 3
        disp('y(t)=t^2')
        figure
        plot(t,z1,t,2*t),title('y(t)=t^2'), legend('z1','2*t')
        figure
        plot(t,z2,t,2),title('y(t)=t^2'), legend('z2','2')
    case 4
        disp('y(t)=sin(t)+eps*sin(1000*t)')
        figure
        plot(t,z1,t,cos(t)),title('y(t)=sin(t)+eps*sin(1000*t)', 'FontWeight','bold')
        legend('z1','cos(t)+1000*eps*cos(1000*t)')
        figure
        plot(t,z2,t,-sin(t)),title('y(t)=sin(t)+eps*sin(1000*t)', 'FontWeight','bold')
        legend('z2','-sin(t)-(1000^2)*eps*sin(1000*t)')
    case 5
        disp('y(t)=sin(t)+\sigma*randn')
        figure
        plot(t,z1,'b',t,cos(t),'r'),title('y(t) = sin(t) + \sigma * randn'),
        legend('z1 (SM)','y' = cos(t)'), xlabel('t'), ylabel('Z1 <-> cos(t)')
        figure
        plot(t,z2,'b',t,-sin(t),'r'),title('y(t) = sin(t) + \sigma * randn'),
        legend('z2 (SM)','y' = -sin(t)'), xlabel('t'), ylabel('Z2 <-> -sin(t)')
    case 6
        disp('y(t)=t^2+sigma^2*randn(1,1)')
        figure
        plot(t,z1,t,2*t),title('y(t)=t^2+sigma*randn(1,1)'), legend('z1','2*t')
        figure
        plot(t,z2,t,2),title('y(t)=t^2+sigma*randn(1,1)'), legend('z2','2')
end

% B - Processing Results for signal with Random noise (case 5 , 6)

% Define new variables:
% delta1 , delta1avg , sigma_delta1
% delta2 , delta2avg , sigma_delta2

N=length(t)-1;
switch choose
    case 5 % y(t)=sin(t)+random_noise
        delta1=z1-cos(t);
        delta1avg=sum(delta1)./N;
        sigma_delta1=sum((delta1-delta1avg).^2)./N;

        delta2=z2+sin(t); % delta2 = z2-(-sin(t))
        delta2avg=sum(delta2)./N;
        sigma_delta2=sum((delta2-delta2avg).^2)./N;
        fprintf('\nProcessing Results for signal with Random Noise: \nsigma_delta1 =
%.2f\nsigma_delta2 = %.2f \n',sigma_delta1,sigma_delta2);
    case 6 % y(t)=t^2+random_noise
        delta11=z1-2*t;
        delta11avg=sum(delta11)./N;
        sigma_delta11=sum((delta11-delta11avg).^2)./N;

        delta22=z2-2;
        delta22avg=sum(delta22)./N;
        sigma_delta22=sum((delta22-delta22avg).^2)./N;
        fprintf('\nProcessing Results for signal with Random Noise: \nsigma_delta1 =
%.2f\nsigma_delta2 = %.2f \n',sigma_delta11,sigma_delta22);
end

```

4.5 מסקנות ביניים ושאלות מנחות

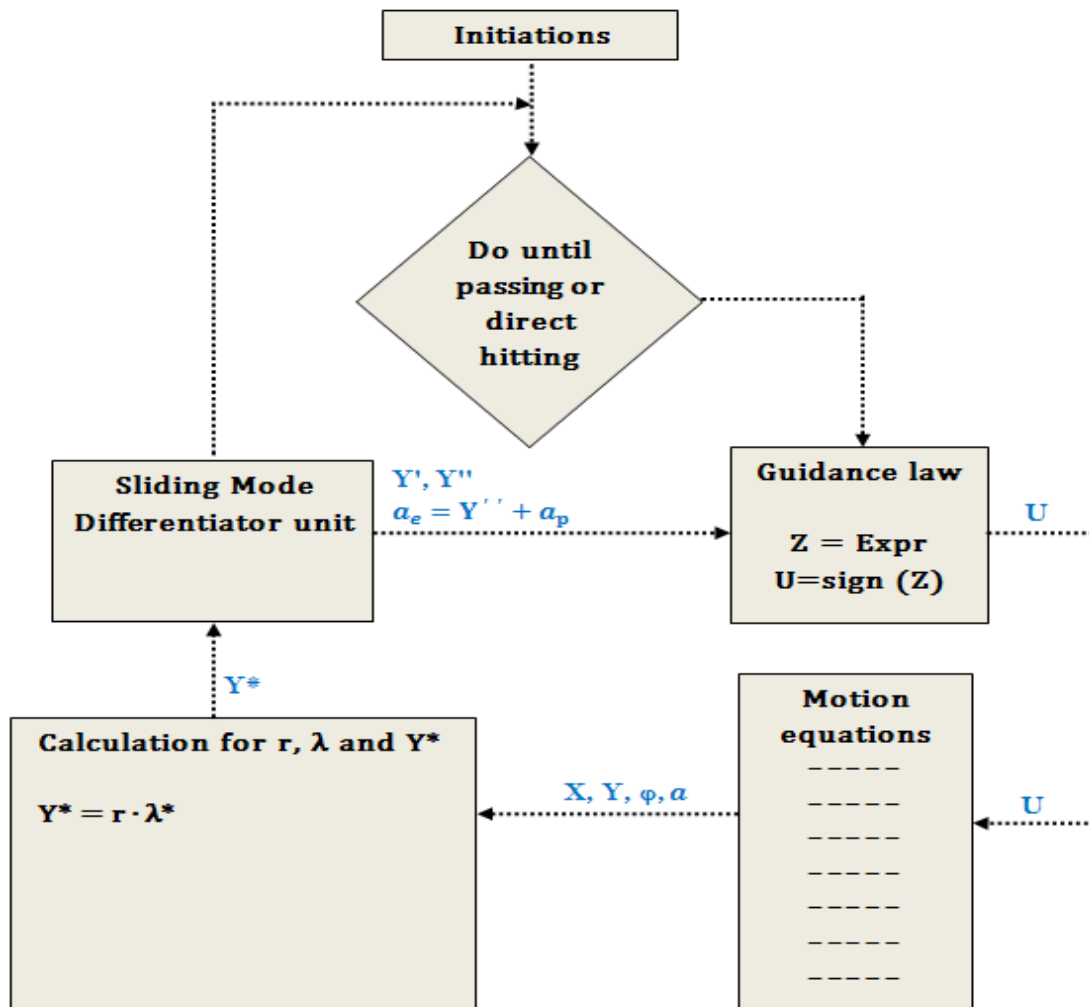
- נוכח בדיקתנו, הסקנו כי לקבועי הגוזר יש חשיבות רבה עבור טיב תוצאות הגזירה.
- בנוסף, ניתן לראות כי קבועים אלו משתנים בהתאם לקלט $y(t)$ של המערכת ושינויים אלו עשויים לנוע סביב סדרת ערכים גדולה.
- מתן קבועים קטנים / גדולים מהדרוש יגררו תוצאות גזירה לא טובות

מסקנות אלו העלו בפנינו שאלות כגון :

- מהם ערכי הקבועים המתאימים וכיצד עלינו לבחור אותם?
- האם ערכים אלו קבועים ואין אפשרות שישתנו בזמן תעופה ובהתאם לשינויי המערכת?
- האם ערכים אלו משתנים ממערכת אחת למערכת אחרת וכיצד הרעש האקראי משפיע על ערכי הקבועים?

4.6 השתלבות הגוזר במערכת הכוללת

את ההשתלבות של הגוזר כמנגנון שחזור לסיוע בקרה, ניתן לתאר דרך תרשים המלבנים הבא :



כפי שצוין קודם לכן, תפקידו של הגוזר במערכת הינו אספקת מידע על נגזרות פונקציית הקלט, במקרה זה $y(t)$ המהווה ביטוי לתנועה במודל זה ככפל של המרחק בין הגופים כפול זווית קו הראיה האופקית λ .

5. קבועי הגזר Differentiation parameters

לאחר אבחון הגזר, הסקנו כי איכות תוצאות גזירתו תלויות במידה ניכרת בערכי הקבועים של המערכת ■ כלומר: $\lambda_0, \lambda_1, \dots, \lambda_n$

מידע קודם מדויק לגבי בחירת ערכים לא הופיע בספרות שנבחנו (במאמרו של Levant נכתב כי ערכים אלו צריכים להיבחר בצורה רקורסיבית, וכי הדרך הטובה ביותר להשיגן היא באמצעות סימולציה ממוחשבת).

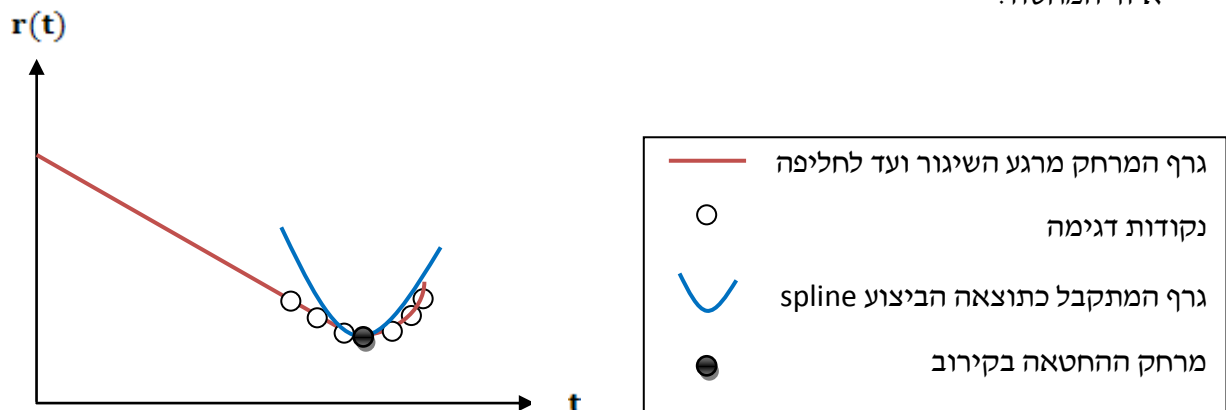
הדרך בה החלטנו לנקוט למציאת ערכי הקבועים של המערכת התחלקה לשני שלבים:

1. יצירת מנגנון (תוכנית) אשר באמצעותה ביצעתי את בדיקת הקריטריונים על סדרות (במקרה של עד נגזרת שנייה מדובר בשלוש) של ערכים רנדומאליים, עד לקבלת מה שהגדרנו כ"סדרה אופטימאלית עבור המערכת הנומריית".
2. תיאור גרף ההתפלגות של *מרחק ההחטאה עבור סדרות קבועים שנמצאו רנדומאליים מטווחים נקובים.

תוצאת התוכנית סיפקו ידע לגבי תחומי ה λ ות הדרושים על מנת למדל היטב את המערכת הכוללת.

מרחק ההחטאה – כזכור הבעיה הינה בעיית יירוט ועל כן מדד מרכזי להצלחת השיגור הוא קבלת מרחק החטאה מינימאלי (סנטימטרים ספורים). בנוסף, מקרה של חליפה (כאשר הטיל הרודף חולף על פני הטיל נרדף) היא החטאה מבחינתנו. באמצעות שיטת spline למרחק היחסי בין הטילים בנקודות שלפני חליפתם עד לכמה נקודות בודדות לאחר החליפה ובלקחת מינימום לפרבולה המתקבלת, יכולנו להסיק את מרחק ההחטאה הסופי.

איור המחשה :



5.1 קוד תוכנית ממומשת – Matlab – Levant_lambda_criterion

```
function Levant_lambda_criterion( v_p , v_e , phi_p0 , phi_e0 , a_p0 , a_e0 , ap_max, ae_max,
r0, n_runs)
%-----
% RUN WITH: >> Levant_lambda_criterion(2300 , 2700 , 0 , 0 , 0 , 0 , 200 , 100 , 20000 , 100)
%-----

clc, close all
fprintf('Generate Cumulative Distribution Function for series of Levant lamdas\n');

%-----
%constants
%-----
sigma=1e-4;
h=4/10000;
tau_p=0.2;
tau_e=0.2;

%-----
%Leavant lamdas (lam0,lam1,lam2)
%-----
lam0=[100 150 200 400 300];
lam1=[1000 800 500 600 300];
lam2=[30 25 20 100 15];
colors=['b','k','r','g','m'];
rand('state',1)
best_series=0;
best_promised_distance=100;

for lam=1:length(lam0)

    fprintf('\nItaration: (%s)  please wait...\n',num2str(lam))

    for k=1:n_runs

        clear t
        t(1)=0;
        i=1;
        z0(1)=0;
        z1(1)=0;
        z2(1)=0;

        %-----
        %Pursuer initial conditions
        %-----
        xp(1)=0;
        yp(1)=0;
        phi_p(1)=phi_p0;
        a_p(1)=a_p0;

        %-----
        %Evader initial conditions
        %-----
        xe(1)=r0;
        ye(1)=0;
        phi_e(1)=phi_e0;
        a_e(1)=a_e0;
        V=1; % Evader control

        r(1)=r0; % initial distance 20[km]
        rr=r0+1;
        t_go=r0/(v_p+v_e);
```

```

while r(i)<rr

    %-----
    % note that:
    % a_e = y'+a_p = z2+a_p
    % t_end = t+t_go = t-r/r_dot
    %-----
    Z = calculateZ_mode(t(i)+t_go,t(i),z0(i),z1(i),a_p(i),z2(i)+a_p(i),tau_p,tau_e);
    U=sign(Z); % Pursuer Control

    %-----Pursuer
    dxpi = v_p*cos(phi_p(i));
    xp(i+1)=xp(i)+h*dxpi;

    dypi = v_p*sin(phi_p(i));
    yp(i+1)=yp(i)+h*dypi;

    dphi_pi = a_p(i)/v_p;
    phi_p(i+1)=phi_p(i)+h*dphi_pi;

    da_pi=(ap_max*U-a_p(i))/tau_p;
    a_p(i+1)=a_p(i)+h*da_pi;
    %-----

    %-----Evader
    dxei=-v_e*cos(phi_e(i));
    xe(i+1)=xe(i)+h*dxei;

    dyei=v_e*sin(phi_e(i));
    ye(i+1)=ye(i)+h*dyei;

    dphi_ei = a_e(i)/v_e;
    phi_e(i+1)=phi_e(i)+h*dphi_ei;

    da_ei=(ae_max*V-a_e(i))/tau_e;
    a_e(i+1)=a_e(i)+h*da_ei;
    %-----

    x=xe(i+1)-xp(i+1); % x = x_e - x_p
    y=ye(i+1)-yp(i+1); % y = y_e - y_p
    x_dot=-v_e*cos(phi_e(i+1))-v_p*cos(phi_p(i+1)); % x'
    y_dot=v_e*sin(phi_e(i+1))-v_p*sin(phi_p(i+1)); % y'

    range=sqrt(x^2+y^2); % current range
    r_dot=(x*x_dot+y*y_dot)/range;% dr/dt
    r(i+1)=range;

    t_go=-range/r_dot;
    if t_go<0
        t_go=1e-6; % avoiding negative numerical "time to go"
    end

    lamda=atan((ye(i)-yp(i))/(xe(i)-xp(i)));
    R=unifrnd(-sigma,sigma);
    lamda_star(i)=lamda+R;

    %-----
    % yi = r * lamda*
    %-----
    yi=r(i)*lamda_star(i);

    %-----
    %Sliding Mode equations
    %-----
    w0=-lam0(lam) * [abs(z0(i)- yi)]^(2/3) * sign(z0(i)- yi) + z1(i);
    z0(i+1)=z0(i) + h*w0;% z0=yi

    w1=-lam1(lam) * abs(z1(i)-w0)^(1/2) * sign(z1(i)-w0) + z2(i);
    z1(i+1)=z1(i) + h*w1;% z1=yi'

    w2=-lam2(lam) * sign(z2(i)-w1);
    z2(i+1)=z2(i) + h*w2;% z2=yi'
    %-----

    rr=r(i);
    t(i+1)=t(i)+h;
    i=i+1;

end %of while

```

```

        distance_for_activation=rr;
        n_t=length(t);
        t1=t(n_t-2);
        t2=t(n_t-1);
        t3=t(n_t);

        t_(1:3)=t(n_t-2:n_t);
        t__=t1:(t3-t1)/100:t3;

        % N_r= index of miss distance in multiple simulation
        missed_distance(k)=min(spline(t_,r(n_t-2:n_t),t__));

%         format short g
%         fprintf('%s) miss distance: %.2f[m]\n', num2str(k), missed_distance(k))

    end %of for

    sorted_missed_distance = sort(missed_distance);

    fprintf('*****\n\nlam0=%s  lam1=%s  lam2=%s\n',...
        num2str(lam0(lam)),num2str(lam1(lam)),num2str(lam2(lam)))
    promised_95percent_distance =

    sorted_missed_distance(floor(0.95*length(missed_distance)));
    fprintf('*****\n\npromised distance (in 95 percent):
%.2f[m]\n',...
        promised_95percent_distance)
    fprintf('*****\n\nAvarage Miss Distance:
%.2f[m]\n*****\n',...
        sum(missed_distance)/n_runs)

    if(promised_95percent_distance < best_promised_distance)
        best_series=lam;
        best_promised_distance=promised_95percent_distance;
    end

    fprintf('\n\n')

    hold on
    plot(sorted_missed_distance,1/n_runs:1/n_runs:1,colors(lam)),

end
legend(' (1) ', ' (2) ', ' (3) ', ' (4) ', ' (5) ')
axis([0 4 0 1]);
xlabel('missed distance'), ylabel('probability')
title('Cumulative Distribution Function','BackgroundColor','y')
fprintf('---- RESULTS ----\n\nBest series: (%s) \nwith promised distance:
%.2f[m]\n',num2str(best_series),best_promised_distance)
hold off

```

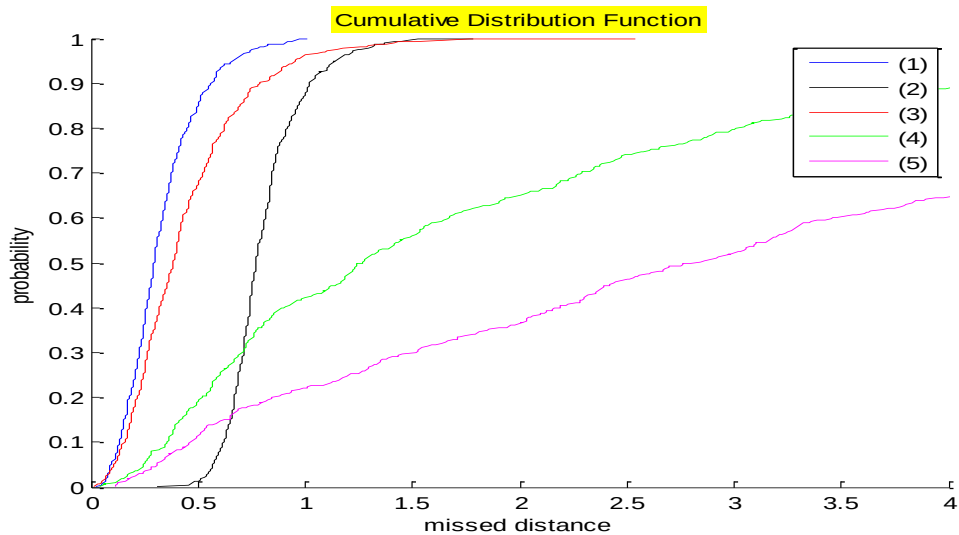
5.2 מטרת התוכנית Levant_lambda_criterion ודרך פעולתה

כפי שמוצג בקוד הנ"ל, ניתנה אפשרות לאבחן את טיב הגוזר בהינתן סדרת קבועים נתונה. הייצוג גרפי של פונקציית ההתפלגות, הניב תמונה חד משמעית וברורה לגבי אותה סדרת ערכים נקובה. כלומר ככל שגרף ההתפלגות ה i היה צמוד יותר לציר האנכי, ניתן להסיק כי סדרת הקבועים ה i טובה יותר. זאת משום שמרחק פגיעה קטן יותר התקבל בהסתברות גבוהה יותר. הרצות אלו בוצעו עבור סדרת 500 איטרציות מונטה קרלו.

בנוסף, נלקחו מדדים נוספים :

- pd - מרחק החטאה מובטח ב 95% מהמקרים promised distance (in 95 percent)
- ad - מרחק ההחטאה הממוצע average miss distance

בסוף האיטרציות הודגשה הסדרה הטובה ביותר (מבין סדרות שנאמדו) והוצגו הפרופילים כפי שניתן לראות כדוגמה בגרף הבא :



דרך הדוגמא הנ"ל, ניתן לראות בחינה עבור 5 סטים של למדות :

טבלת הקלט פלט ומקרא :

מסקנות ומדדים	צבע גרף	סדרת למדות
מבחינת הצמדה לציר ה y (ערכי x) : תוצאות טובות pd=0.96[m] , ad=0.44[m]	—	$\lambda_0 = 200 \lambda_1 = 700 \lambda_2 = 40$
מבחינת הצמדה לציר ה y (ערכי x) : תוצאות לא טובות pd=5.17[m] , ad=1.84[m]	—	$\lambda_0 = 500 \lambda_1 = 500 \lambda_2 = 100$
מבחינת הצמדה לציר ה y (ערכי x) : תוצאות בינוניות pd=1.14[m] , ad=0.8[m]	—	$\lambda_0 = 100 \lambda_1 = 1400 \lambda_2 = 30$
מבחינת הצמדה לציר ה y (ערכי x) : תוצאות מעולות pd=0.65[m] , ad=0.28[m]	—	$\lambda_0 = 100 \lambda_1 = 1000 \lambda_2 = 30$
מבחינת הצמדה לציר ה y (ערכי x) : תוצאות גרועות pd=7.15 [m] , ad=3.15 [m]	—	$\lambda_0 = 300 \lambda_1 = 300 \lambda_2 = 15$

בחינותינו בוצעו עבור סדרות רבות (אקראיות) של למדות. בדוגמא זו ניתן לראות כי הסדרה הטובה ביותר היא הסדרה הרביעית שניתנה כפלט.

6. הרחבת המודל עבור λ ות משתנות בזמן – רעיון חדש

6.1 מוטיבציה ליצירת מנגנון "הסתגלות"

המוטיבציה ליצירת מנגנון שיאפשר את שינוי ערכי הקבועים תוך כדי הסימולציה נבעה מעצם הגדרת המערכת. המערכת הינה מערכת דינאמית, המשתנה בזמן, אשר מושפעת מגורמים רבים. אי לכך ובהתאם לזאת, אולי ישנו צורך למתן אפשרות דינאמיקה גם עבור הגוזר, כלומר יצירת מנגנון "הסתגלות" עם השינוי בזמן.

שיטה זו לא הופיעה כלל במאמרו של A. Levant והיא הצעה שהועלתה במהלך החקר, אשר נבחנה על ידינו והוכיחה נכונותה.

6.2 תבנית $\lambda_i(t)$ נבחרות עבור המערכת PE

הרעיון היה ליצור אפשרות ל λ ות להשתנות בזמן. נבחרה תבנית מהצורה:

$$\lambda_0(t) = \lambda_0 \cdot (1 - k_{\lambda_0} \cdot t)$$

$$\lambda_1(t) = \lambda_1 \cdot (1 - k_{\lambda_1} \cdot t)$$

$$\lambda_2(t) = \lambda_2 \cdot (1 - k_{\lambda_2} \cdot t)$$

כאשר $k_{\lambda_0}, k_{\lambda_2}, k_{\lambda_1}$ הינם קבועים אשר ניתנים לתמרון על ידינו. מעצם הגדרת התבנית ניתן לראות כי עבור $k_{\lambda_i} = 0, i = 1, 2, \dots, n$ נקבל λ_i קבוע לאורך כל הסימולציה.

6.3 בחינת ההשערה על מערכת PE

בכדי לבדוק השערותנו, נכתבה *תוכנית שבה הוצבו התבניות, ושם השוונו בין שני מקרים, מקרה ראשון בו הקבועים שניתנו כקלט היו ללא שינוי, כלומר נשארו קבועים ומקרה נוסף בו "קבועי הגוזר" שונו עם הזמן לפי פונקציות ההשתנות שהוגדרו על ידינו.

*שם התוכנית: `pursuer_evader_complete_view_with_ChangeInTime_lamdas`

6.3.1 דרך המבחן

עבור אותו מספר שיגורים (סימולציות מונטה קרלו) ובהינתן אותם קבועים בזמן ההתחלתי $\lambda_0(t) = a, \lambda_1(t) = b, \lambda_2(t) = c$ בוצעו שתי הרצות.

6.3.1.1 הרצה ראשונה

בוצעה הרצה כאשר $k_{\lambda_i} = 0, i = 1, 2, 3$

6.3.1.2 הרצה שנייה

בוצעה כאשר לפחות אחד מה λ ות משתנה בזמן (בדוגמה שבהמשך שלושתן משתנות).

לאחר מכן בחנו טיב פגיעה באמצעות פונקצית ההתפלגות של מרחק ההחטאה והמדדים הנוספים אשר הוזכרו בסעיף 5.

6.3.2 מימוש המבחן על המערכת PE, דוגמא

פרמטרים להרצה ראשונה :

$$\begin{aligned}\lambda_0(t) &= 100 \\ \lambda_1(t) &= 100 \\ \lambda_2(t) &= 50 \\ k_{\lambda_0} &= k_{\lambda_2} = k_{\lambda_1} = 0\end{aligned}$$

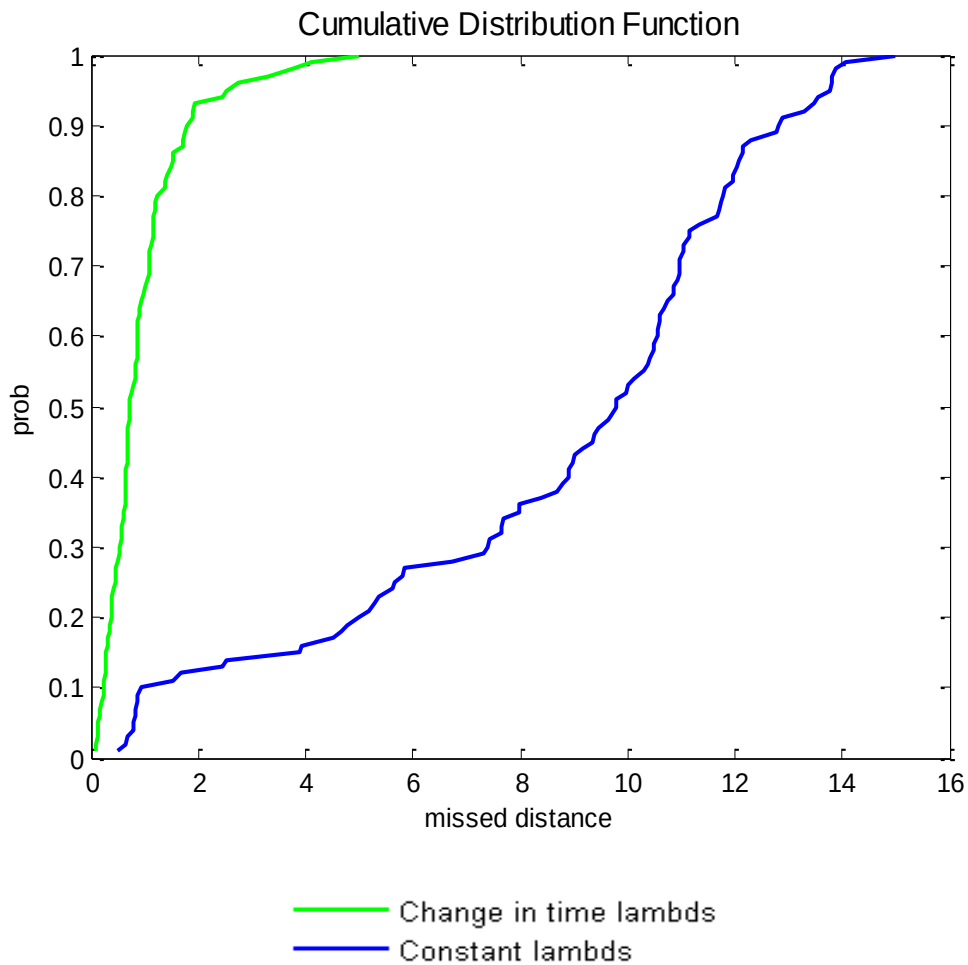
פרמטרים להרצה שנייה :

$$\begin{aligned}\lambda_0(t) &= 100 \\ \lambda_1(t) &= 100 \\ \lambda_2(t) &= 50 \\ k_{\lambda_0} &= k_{\lambda_2} = -1, k_{\lambda_1} = -2\end{aligned}$$

6.3.3 טבלת תוצאות השוואתית בין שתי ההרצות

הערות	λ ות משתנות בזמן	λ ות קבועות	
מספר סימולציות זהה	100	100	מספר שיגורים (n)
בדיקה עבור התנהגות המערכת עם λ ות משתנות בזמן אל מול λ ות קבועות	$\lambda_0 = 100 \cdot (1 - (-1) \cdot t)$ $\lambda_1 = 100 \cdot (1 - (-2) \cdot t)$ $\lambda_2 = 50 \cdot (1 - (-1) \cdot t)$	$\lambda_0 = 100$ $\lambda_1 = 100$ $\lambda_2 = 50$	ערכי λ ות
מרחק החטאה מובטח קטן משמעותית	2.51[m]	13.77[m]	מרחק החטאה מובטח עבור 95% מהמקרים
מרחק ההחטאה הממוצע ירד ממרחק של קרוב ל 9 מטרים לפחות ממטר!	0.97[m]	8.57[m]	מרחק החטאה ממוצע עבור n שיגורים

6.3.4 השוואת גרפי התפלגות המצטברת בין שתי ההרצות



ניתן לראות בבירור כי עבור לוח המשתנות בזמן למערכת הניתנת יש תוצאות טובות בהרבה. כמובן שגם מרחק החטאה מובטח ב 95% מהמקרים ו מרחק החטאה הממוצע השתפרו במידה לא מבוטלת.

6.3.5 מסקנות המבחן ורעיונות להמשך

- ישנם מקרים (כבר בניסויים שנערכו נמצאו כמה כאלו) בהם בהינתן ערכי קבועי התחלה של הגוזר למערכת, תוצאות פגיעה מוצלחות יותר מתקבלות בעת שינוי הקבועים עם השתנות הזמן וזאת ע"י פונקציית שינוי מוגדרת מראש.
- ההנחה כי עבור המערכת בעלת פונקציית קלט לא ידועה ובעלת תכונות לא ידועות (האות $y(t)$ תלוי בהרבה מאוד גורמים, ומתקבל מפתרון של מערכת משוואות דיפרנציאליות) יהיה יעיל יותר במקרים מסוימים לשנות את קבועי הגוזר בזמן ריצה, דבר שיניב תוצאות פגיעה טובות יותר מאשר אי שינוי הקבועים.
- רעיון נוסף הוא ליצור תמרון של ערכי ה לוח שלא תלוי רק בזמן, אלא גם בגורמי מפתח נוספים במערכת כדוגמת המרחק בין הטיילים, או זווית קו הראיה האופקית. לשם כך דרושה סימולציה מורכבת הרבה יותר ומידע מערכתי רב.

7. הדמיות שיגור

כחלק מביצוע הפרויקט, נכתבה התוכנית `pursuer_evader_movie` אשר באמצעותה יכולנו להציג הדמיות של מקרי שיגור שונים כתלות בכל מרכיבי המערכת. כמו כן, הוגדרה יכולת נוספת עבור הנרדף – היכולת לשנות את פונקציית הבקרה העצמית (שכזכור במצב המוצא נלקחה בצורה הפשוטה ביותר $V(t) = 1$) בזמן ריצה. דבר שגרם לתוצאות שונות: כאלו שעדיין אפשרו תפיסה וכאלו שגרמו להחטאת המטרה.

7.1 מספר דוגמאות הדמייה תחת תנאים שונים

חלק מתוצאות הדמייה ניתן לראות בקבצים המצורפים לפרויקט זה. להלן קישורים, לחיצה כפולה לפתיחת סרטון (במידה ולא נפתח - הסרטונים מצורפים לתיקיה)

הדמיית שיגור של בעיית רודף נרדף בהינתן נתוני המקור:



`pursuer_evader_orig
wmv.`

הדמיית שיגור של בעיית רודף נרדף כאשר בחירת הקבועים בגוזר ההחלקה אינה נכונה:



`pursuer_evader_inco
rrect_lambds.wmv`

הדמיית שיגור של בעיית רודף נרדף כאשר:

פונקציית הבקרה V משתנה באופן יזום, לאחר 2 שניות מרגע השיגור לתבנית $V = |\tan(t)|$



`pursuer_evader_a.w
mv`

הדמיית שיגור של בעיית רודף נרדף כאשר:

פונקציית הבקרה V משתנה באופן יזום, לאחר 3 שניות מרגע השיגור לתבנית $V = t^2$



`pursuer_evader_f.w
mv`

7.2 קוד תוכנית מממשת – Matlab – pursuer_evader_movie

```
function pursuer_evader_movie( v_p , v_e , phi_p0 , phi_e0 , a_p0 , a_e0 , ap_max, ae_max,
r0)
%-----
% RUN WITH: >> pursuer_evader_movie(2300 , 2700 , 0 , 0 , 0 , 0 , 200 , 100 , 20000)
%-----
clc; close all

h=4/10000;
sigma=1e-4;
tau_p=0.2;
tau_e=0.2;
t(1)=0;
i=1;
z0(1)=0;
z1(1)=0;
z2(1)=0;
call = 0;

%-----
%Leavant lamdas (lam0,lam1,lam2)
%-----
%%Direct Hit (BOOM)
lam0=100;
lam1=1000;
lam2=30;

%-----
%" optimal Levant lamdas " for the numeric system
%-----
%[lam0, lam1, lam2, call] = optimal_Levant_lamdas_for_numeric_system(1e-4);

set(0, 'DefaultFigureColor', [1 0.97 1])

if (call==1)
    to_create_movie_simulation=input('Q: to continue and create movie simulation ? (y / n):
    ', 's');
    if (to_create_movie_simulation == 'y')
        close all
    else
        return
    end
end

%-----
%Pursuer initial conditions
%-----
xp(1)=0;
yp(1)=0;
phi_p(1)=phi_p0;
a_p(1)=a_p0;

%-----
%Evader initial conditions
%-----
xe(1)=r0;
ye(1)=0;
phi_e(1)=phi_e0;
a_e(1)=a_e0;
V=1; % Evader control.
%try: V=2
%V=2; %, if (t(i)>=1.5) v=0.2 end %b

r(1)=r0; % initial distance 20[km]
rr=r0+1;
t_go=r0/(v_p+v_e);

rand('state',1)
fig=figure;

aviobj = avifile('pursuer_evader_incorrect_lambds3.avi','compression','None');
my_axis=[0 20000 0 1000];
```

```

while r(i)<rr

    % V=exp(sin(t(i))); % Evader control
    %-----
    % V - the evader control,
    % can be changed in time
    %-----
    %     if t(i)>=2 && t(i)<2.5
    %         V=-1;
    %     end
    %
    %     if t(i)>=2.5 && t(i)<3
    %         V=3;
    %     end
    %
    %     if t(i)>=3
    %         V=-1;
    %     end

    %     if t(i)>=3
    %         %V=-1; %t(i)>=2           %a
    %         %v=0.2;                %b
    %         %V=abs(tan(t(i))); %t(i)>=2 %a
    %         %V=t(i)^2; %t(i)>=3       %f
    %         %V=(-1)*t(i)^2; %t(i)>=3  %d
    %         %V=-2*t(i)+4;           %c
    %         %V=1; %t(i)>=2 and V.start=1/10 %a
    %     end

    %-----
    % a_e = y''+a_p = z2+a_p
    % t_end = t+t_go = t-r/r_dot
    %-----
    Z = calculateZ_mode(t(i)+t_go,t(i),z0(i),z1(i),a_p(i),z2(i)+a_p(i),tau_p,tau_e);
    U=sign(Z); % Pursuer Control

    %-----Pursuer
    dxpi = v_p*cos(phi_p(i));
    xp(i+1)=xp(i)+h*dxpi;

    dypi = v_p*sin(phi_p(i));
    yp(i+1)=yp(i)+h*dypi;

    dphi_pi = a_p(i)/v_p;
    phi_p(i+1)=phi_p(i)+h*dphi_pi;

    da_pi=(ap_max*U-a_p(i))/tau_p;
    a_p(i+1)=a_p(i)+h*da_pi;
    %-----

    %-----Evader
    dxei=-v_e*cos(phi_e(i));
    xe(i+1)=xe(i)+h*dxei;

    dyei=v_e*sin(phi_e(i));
    ye(i+1)=ye(i)+h*dyei;

    dphi_ei = a_e(i)/v_e;
    phi_e(i+1)=phi_e(i)+h*dphi_ei;

    da_ei=(ae_max*V-a_e(i))/tau_e;
    a_e(i+1)=a_e(i)+h*da_ei;
    %-----

    x=xe(i+1)-xp(i+1); % x = x_e - x_p
    y=ye(i+1)-yp(i+1); % y = y_e - y_p
    x_dot=-v_e*cos(phi_e(i+1))-v_p*cos(phi_p(i+1)); % x'
    y_dot=v_e*sin(phi_e(i+1))-v_p*sin(phi_p(i+1)); % y'

    range=sqrt(x^2+y^2); % current range
    r_dot=(x*x_dot+y*y_dot)/range;% dr/dt
    r(i+1)=range;

    t_go=-range/r_dot;
    if t_go<0
        t_go=1e-6; % avoiding negative numerical "time to go"
    end

    lamda=atan((ye(i)-yp(i))/(xe(i)-xp(i)));
    R=unifrnd(-sigma,sigma);
    lamda_star(i)=lamda+R;

```

```

%-----
% yi = r * lamda*
%-----
yi=r(i)*lamda_star(i);

%-----
%Sliding Mode equations
%-----
w0=-lam0 * [abs(z0(i)- yi)]^(2/3) * sign(z0(i)- yi) + z1(i);
z0(i+1)=z0(i) + h*w0;% z0=yi

w1=-lam1 * abs(z1(i)-w0)^(1/2) * sign(z1(i)-w0) + z2(i);
z1(i+1)=z1(i) + h*w1;% z1=yi'

w2=-lam2 * sign(z2(i)-w1);
z2(i+1)=z2(i) + h*w2;% z2=yi'
%-----

%-----
%Create Movie Animation
%-----
k=mod(i,400);

if(i==1 || k==0)

    plot(xp(i),yp(i),'rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.2 .5
.9],'MarkerSize',10)
    xlabel('x[m]','FontWeight','b'), ylabel('y[m]','FontWeight','b')
    hold on

plot(xe(i),ye(i),'rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',1
0)
    plot([xp(i),xe(i)],[yp(i),ye(i)],':k','LineWidth',1)
    axis(my_axis), legend('Pursuer','Evader');

    distance=['distance = ' num2str(r(i), '% 10.2f') ' [m]'];
    ttg=['time to go = ' num2str(t_go, '% 10.2f') ' [sec]'];
    title({distance;ttg},'BackgroundColor','y');

    F = getframe(fig);

    aviobj = addframe(aviobj,F);

    hold off

end
%-----

rr=r(i);
t(i+1)=t(i)+h;
i=i+1;

end %of while

%-----
%find missed distance by spline
%-----
n_t=length(t);
t1=t(n_t-2);
t2=t(n_t-1);
t3=t(n_t);
t_(1:3)=t(n_t-2:n_t);
t__=t1:(t3-t1)/100:t3;
missed_distance = min(spline(t_,r(n_t-2:n_t),t__));
plot(xp(end),yp(end),'o','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.2 .5
.9],'MarkerSize',16)
xlabel('x[m]','FontWeight','b'), ylabel('y[m]','FontWeight','b')

hold on

plot(xe(end),ye(end),'s','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize
',10)
xlabel('x[m]'), ylabel('y[m]')

plot(xp,yp,'b','LineWidth',2)
hold on
plot(xe,ye,'r','LineWidth',2)
xlabel('x[m]','FontWeight','b'), ylabel('y[m]','FontWeight','b')
axis(my_axis), legend('Pursuer','Evader');
distance=['final distance = ' num2str(missed_distance,'%10.2f') ' [m]'];
title({distance,'time to go = 0 [sec]'},'fontsize',12,'BackgroundColor','y','FontWeight','b')

```

```

%-----
% yi = r * lamda*
%-----
yi=r(i)*lamda_star(i);

%-----
%Sliding Mode equations
%-----
w0=-lam0 * [abs(z0(i)- yi)]^(2/3) * sign(z0(i)- yi) + z1(i);
z0(i+1)=z0(i) + h*w0;% z0=yi

w1=-lam1 * abs(z1(i)-w0)^(1/2) * sign(z1(i)-w0) + z2(i);
z1(i+1)=z1(i) + h*w1;% z1=yi'

w2=-lam2 * sign(z2(i)-w1);
z2(i+1)=z2(i) + h*w2;% z2=yi'
%-----

%-----
%Create Movie Animation
%-----
k=mod(i,400);

if(i==1 || k==0)

    plot(xp(i),yp(i),'rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.2 .5
.9],'MarkerSize',10)
    xlabel('x[m]','FontWeight','b'), ylabel('y[m]','FontWeight','b')
    hold on

plot(xe(i),ye(i),'rs','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',10)
)

plot([xp(i),xe(i)],[yp(i),ye(i)],':k','LineWidth',1)
axis(my_axis), legend('Pursuer','Evader');

distance=['distance = ' num2str(r(i), '% 10.2f') ' [m]'];
ttg=['time to go = ' num2str(t_go, '% 10.2f') ' [sec]'];
title([distance;ttg],'BackgroundColor','y');

F = getframe(fig);

aviobj = addframe(aviobj,F);

hold off

end
%-----

rr=r(i);
t(i+1)=t(i)+h;
i=i+1;

end %of while

%-----
%find missed distance by spline
%-----
n_t=length(t);
t1=t(n_t-2);
t2=t(n_t-1);
t3=t(n_t);
t_(1:3)=t(n_t-2:n_t);
t__=t1:(t3-t1)/100:t3;
missed_distance = min(spline(t_,r(n_t-2:n_t),t__));
plot(xp(end),yp(end),'o','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor',[.2 .5
.9],'MarkerSize',16)
xlabel('x[m]','FontWeight','b'), ylabel('y[m]','FontWeight','b')

hold on

plot(xe(end),ye(end),'s','LineWidth',2,'MarkerEdgeColor','k','MarkerFaceColor','r','MarkerSize',
10)
xlabel('x[m]'), ylabel('y[m]')

plot(xp,yp,'b','LineWidth',2)
hold on
plot(xe,ye,'r','LineWidth',2)
xlabel('x[m]','FontWeight','b'), ylabel('y[m]','FontWeight','b')
axis(my_axis), legend('Pursuer','Evader');
distance=['final distance = ' num2str(missed_distance,'%10.2f') ' [m]'];
title([distance,'time to go = 0 [sec]'],'fontSize',12,'BackgroundColor','y','FontWeight','b')

```



```

if (missed_distance<1)

plot(xe(end),ye(end),'o','LineWidth',2,'MarkerEdgeColor','r','MarkerFaceColor','y','MarkerSize',
10)
axis(my_axis), legend('Pursuer','Evader');

annotation('textbox',[.15,.2,.2,.065],'String','Stroke! (BOOM)','FontSize',13,'FitBoxToText','on',
'BackgroundColor','y')
elseif(missed_distance>=1 && missed_distance<5)
plot([xp(i),xe(i)],[yp(i),ye(i)],':k','LineWidth',1)
annotation('textbox',[.15,.2,.2,.065],'String','Almost
Stroke','FontSize',13,'FitBoxToText','on','BackgroundColor','y')
elseif(missed_distance>=5 && missed_distance<10)
plot([xp(i),xe(i)],[yp(i),ye(i)],':k','LineWidth',1)
annotation('textbox',[.15,.2,.2,.065],'String','Missed By Few
Meters','FontSize',13,'FitBoxToText','on','BackgroundColor','y')
else
plot([xp(i),xe(i)],[yp(i),ye(i)],':k','LineWidth',1)
annotation('textbox',[.15,.2,.2,.065],'String','Pursuer Missed The
Target..','FontSize',13,'FitBoxToText','on','BackgroundColor','y')
end

hold off

for i=1:20
F = getframe(fig);
aviobj = addframe(aviobj,F);
end

close(fig);
aviobj = close(aviobj);

```

8. סיכום

בראשית המסמך הוצגו מושגי בקרה ובעיית יירוט.

מטרת העבודה הייתה לראות את השימוש בגוזר ההחלקה (Sliding Mode Differentiator)

בחוקי הנחיית טילים וזאת דרך המערכת של רודף נרדף PE.

מטרה זו הושגה במספר שלבים החל מהכרת הגוזר בצורתו הכללית דרך השימוש בו, השתלבותו במערכת הכוללת וכלה ביצירת הדמיות שיגור שונות במישור הדו ממדי. הדרך להשגת המטרה עוררה צורך בחקר של מאפייני ייסוד בגוזר ובמערכת, כאשר העיקרי שבהם הוא חשיבות קבועי הגוזר.

נקודה זו הורחבה על ידי ייצור כלי עזר לניתוח כגון: מנגנון ייצור קבועים בצורה אקראית וביצוע אופטימיזציה למערכת PE עם אותם קבועים, אשר טיבם נבחן דרך אמידת מאפייני טיב הגזירה שהגדרנו מעלה.

הרעיון החדש שהוצע, הוא רעיון אשר לא הופיע במאמרו של Levant והוא מן הרחבה נוספת ליכולתו של הגוזר. הצורך נבע מעצם הגדרת הבעיה כבעיה דינאמית ומכך ש"קבועי הגוזר" לא חייבים להיות קבועים, וכי יש לתת פונקציית השתנות (כרגע כביטוי ליניארי) אשר תאפשר שינוי ערכיהם תוך כדי תעופה, פעולה שהוכיחה עצמה כנכונה במקרי יירוט מסוימים. בשלב זה רוכזו מדדי טיב הגזירה בטבלה השוואתית אשר המחישה את הצורך ב λ ות משתנות.

9. ביבליוגרפיה

1. Turetsky, V., and Shinar, J., 'Missile guidance laws based on pursuit-evasion game formulations', *Automatica*, Vol. 39, No. 4, 2003, pp. 607 – 618.
2. Levant, A., "Higher-order sliding modes, differentiation and output-feedback Control", *International Journal of Control*, Vol. 76, No. 9 – 10, 2003, pp. 924 – 941. special issue on Sliding-Mode Control
3. Bartolini et al.: 'Output tracking control of uncertain nonlinear second order systems', *Automatica*, 1997 , 33(12) pp.2203-2212
4. Bartolini, G., Pisano, A., Punta, E., and Usai, E.: 'A survey of applications of second-order sliding mode control to mechanical systems', *Int. J. Control*, 2003, 76, pp. 875–892