רובוטים ניידים| רטוב 1

להב ברק , ת״ז : 312148307 | ירון הלה , ת"ז : 066077033

21 במאי 2023

שאלה 1

חישוב השגיאה מתבצע באופן דומה לחישוב שנראה בתרגול :

$$e = \begin{pmatrix} e_{at} \\ e_{ct} \end{pmatrix} = \begin{pmatrix} \cos\theta_{ref} & \sin\theta_{ref} \\ -\sin\theta_{ref} & \cos\theta_{ref} \end{pmatrix} \begin{pmatrix} x - x_{ref} \\ y - y_{ref} \end{pmatrix} = \begin{pmatrix} \cos\theta_{ref}\left(x - x_{ref}\right) + \sin\theta_{ref}\left(y - y_{ref}\right) \\ \cos\theta_{ref}\left(y - y_{ref}\right) - \sin\theta_{ref}\left(x - x_{ref}\right) \end{pmatrix}$$

ערכי x_{ref}, y_{ref}, x, y נתונים, לכן נותר רק לחשב את θ_{ref} - זה מבוצע ע"י חישוב הזווית של הוקטור המחבר בין הנקודה הבאה לנקודה הנוכחית.

פונקציית חישוב השגיאה מחשבת בנוסף גם את הנגזרת לשגיאה ואת אינטגרל השגיאה, ושומרת אותם בשדות חדשים מתאימים, כך שניתן יהיה להיעזר בהם בפונקציות הבקרים (בפרט PID)

בנוסף חישבנו מהירות רגעית ע"י גזירה בעזרת הפרשים אחוריים של המיקום של הרובוט.

שאלה 2

סעיף 1

מודל חד אופן :

$$u = \begin{pmatrix} v \\ \eta \end{pmatrix}$$

$$\dot{\zeta} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v\cos\theta \\ v\sin\theta \\ \eta \end{pmatrix}$$

כדי לחשב בקר PD נגזור את וקטור השגיאה ונקבל :

$$\dot{e} = \begin{pmatrix} \dot{x}\cos\theta_{ref} + \dot{y}\sin\theta_{ref} \\ \dot{y}\cos\theta_{ref} - \dot{x}\sin\theta_{ref} \end{pmatrix}$$

נציב את המודל, ונקבל

$$\dot{e} = \begin{pmatrix} v\cos\theta\cos\theta_{ref} + v\sin\theta\sin\theta_{ref} \\ v\sin\theta\cos\theta_{ref} - v\cos\theta\sin\theta_{ref} \end{pmatrix} = \begin{pmatrix} v\cos\left(\theta - \theta_{ref}\right) \\ v\sin\left(\theta - \theta_{ref}\right) \end{pmatrix} = \begin{pmatrix} v\cos e_\theta \\ v\sin e_\theta \end{pmatrix}$$

נרצה ששגיאת e_{at} תבקר את המהירות, ושגיאת e_{ct} תבקר את הזווית, לכן הבקר שקיבלנו :

$$u = -\begin{pmatrix} K_{p,at}e_{at} + K_{d,at}v\cos e_\theta + K_{i,at}\int e_{at} \\ K_{p,ct}e_{ct} + K_{d,ct}v\sin e_\theta + K_{i,ct}\int e_{ct} \end{pmatrix}$$

מאחר וקיימת בעיה לחרוג ממהירות בסימולציה, הגבלנו מלאכותית את המהירות - מה שגרם לניוון אפקטיבי של בקר e_{at}.
בעקבות כך השתמשנו בערך K_p גבוה, כדי לזרוק את המהירות למקסימום כמה שיותר מהר - אפקטיבית - יצרנו בקר $bang\, bang$

סעיף 2

מודל אקרמן:

$$u = \begin{pmatrix} v \\ \delta \end{pmatrix}$$

$$\dot{\zeta} = \begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{pmatrix} = \begin{pmatrix} v\cos\theta \\ v\sin\theta \\ \frac{v}{d}\tan\left(\delta\right) \end{pmatrix}$$

נציב למטריצת נגזרת השגיאה ונקבל:

$$\dot{e} = \begin{pmatrix} v\cos\theta\cos\theta_{ref} + v\sin\theta\sin\theta_{ref} \\ v\sin\theta\cos\theta_{ref} - v\cos\theta\sin\theta_{ref} \end{pmatrix} = \begin{pmatrix} v\cos\left(\theta - \theta_{ref}\right) \\ v\sin\left(\theta - \theta_{ref}\right) \end{pmatrix} = \begin{pmatrix} v\cos e_\theta \\ v\sin e_\theta \end{pmatrix}$$

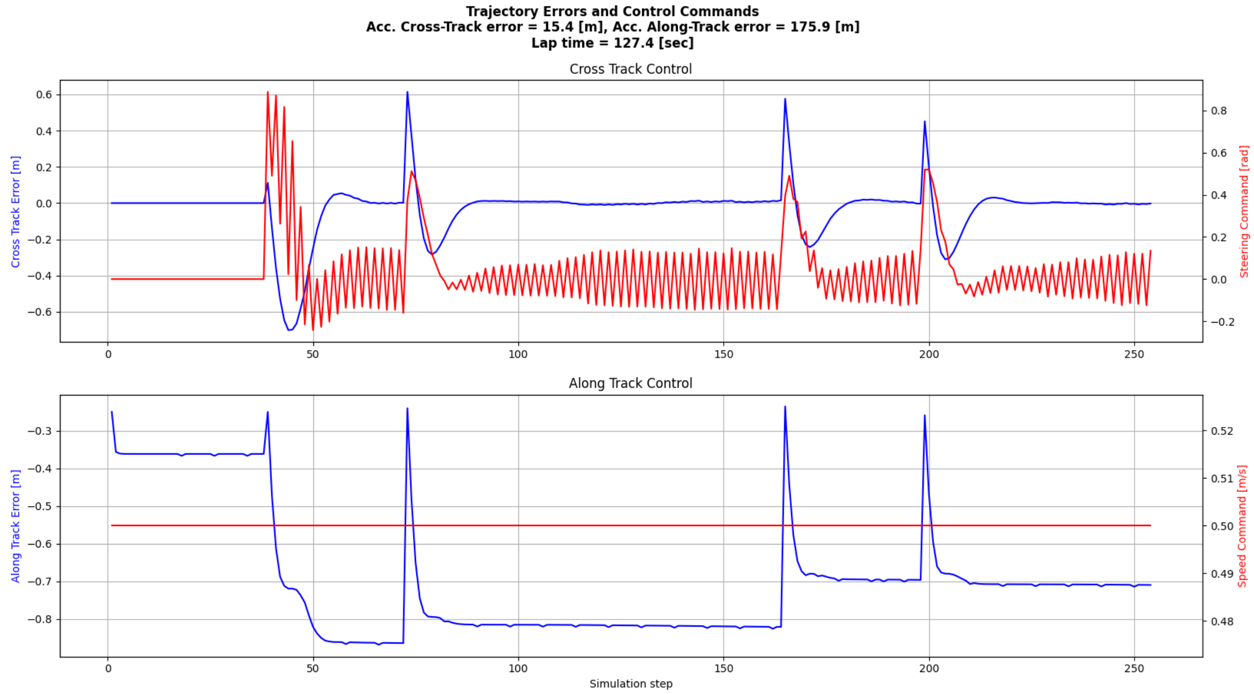
לכן באופן דומה לסעיף 1 נרצה ששגיאת e_{at} תבקר את המהירות, ושגיאת e_{ct} תבקר את הזווית, לכן הבקר שקיבלנו:

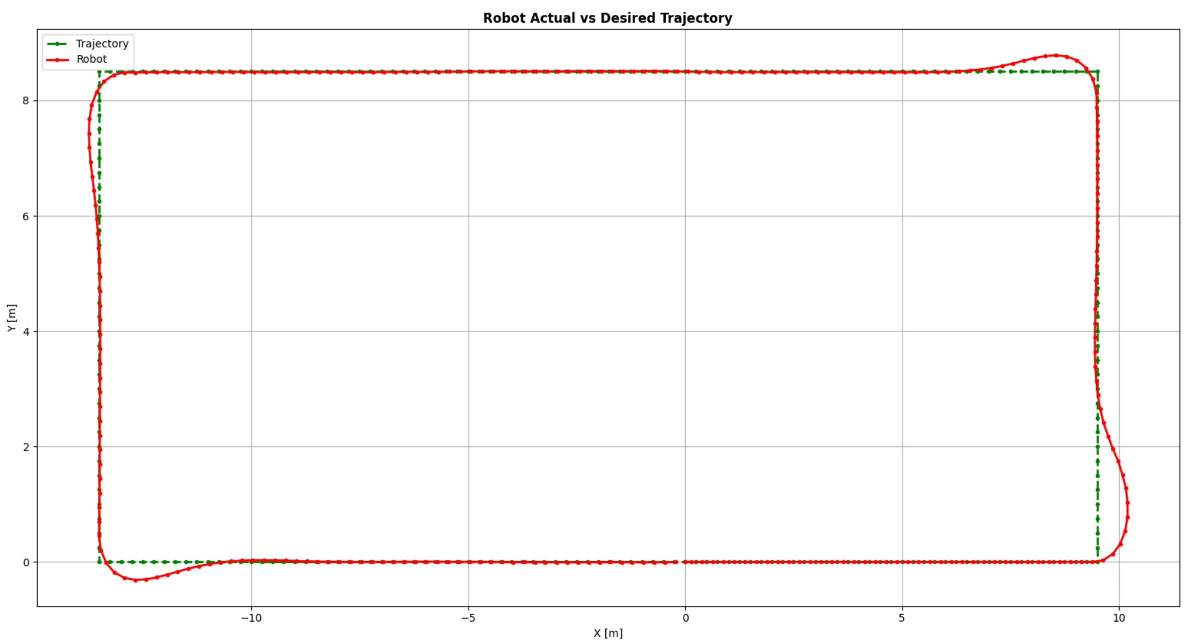
$$u = -\begin{pmatrix} K_{p,at}e_{at} + K_{d,at}v\cos e_\theta + K_{i,at}\int e_{at} \\ K_{p,ct}e_{ct} + K_{d,ct}v\sin e_\theta + K_{i,ct}\int e_{ct} \end{pmatrix}$$

כלומר, הבקר שקיבלנו זהה, מאחר והמודלים זהים עד כדי $\dot{\theta}$, אבל משתנה המצב הזה לא מופיע ישירות בנוסחאות השגיאה.

גם כאן הגבלנו מלאכותית את המהירות - מה שגרם לניוון של בקר e_{at}.

תוצאות הבקר שקיבלנו:


Trajectory Errors and Control Commands
Acc. Cross-Track error = 15.4 [m], Acc. Along-Track error = 175.9 [m]
Lap time = 127.4 [sec]
Cross Track Control
Cross Track Error [m]
Steering Command [rad]
Along Track Control
Along Track Error [m]
Speed Command [m/s]
Simulation step


Robot Actual vs Desired Trajectory
Trajectory
Robot
Y [m]
X [m]

שאלה 3

סעיף 1

used Lookahead 5 as starting reference point, then tested in descending order from 10 to 1 to optimize and observe changes in behavior:

Lookahead Distance	Lap Time [sec]	Total e_{at} $[m]$	Total e_{ct} $[m]$	Notes
10	N/A	N/A	N/A	crashed into 1st corner
9	N/A	N/A	N/A	crashed into 2nd corner
8	123.6	191.1	13.9	grazed 1st corner, still completed lap
7	124.0	158.4	10.6	**best lookahead to optimize lap time**
6	124.5	125.7	8.0	
5	125.0	92.5	6.2	
4	125.5	52.4	5.4	
3	126.0	20.2	4.8	**best lookahead to optimize total errors**
2	126.5	67.2	5.3	low lookahead started harming performance
1	127.0	122.3	7.2	low lookahead greatly harming performance

We can see that starting from lookahead 7 - decreasing lookahead harms lap time linearly (extra 0.5 sec per lookahead decrease), but improves total errors greatly, until we reach lookahead 2 - which is too low and causes large overshoots. Therefore, if we're trying to optimize for lowest accumulated error - the best choice will be lookahead 3, which provides the best tracking

However, if we're trying to optimize for fastest lap time - the best choice seems to be 8, but due to close proximity to the corner, safety margins considerations will lead us to pick 7.

סעיף 2

בסעיף זה מומש בקר $iLQR$. נזכיר שוקטור משתנה המצב X ופקודות הבקרה U הן מהצורה:

$$X = \begin{pmatrix} \mathrm{x} \\ y \\ \theta \end{pmatrix}, U = \begin{pmatrix} v \\ \delta \end{pmatrix}$$

כאשר וקטור X מורכב מהאיברים הבאים (ע"פ הסדר): מקום x, מקום y וכיוון הרכב במערכת צירים אינרציאלית. וקטור U מכיל (ע"פ הסדר) את מהירות הגלגלים ואת זווית הפנייה של הגלגלים (במודל אקרמן) במערכת הצירים של הרכב.

נזכיר שהפונקציה הלא-ליניארית $f\left(X_t, U_t\right)$ המתארת את מודל אקרמן היא מהצורה:

$$f_x\left(X_t,U_t\right)=vcos\theta$$
$$f_y\left(X_t,U_t\right)=vsin\theta$$
$$f_\theta\left(X_t,U_t\right)=\frac{v}{d}tg\delta$$

כאשר d הינו הפרמטר הגיאומטרי של ה-$wheelbase$ של הרכב (נלקח מתוך הקבצים של ה-SOR).
לטובת אופטימיזציית המסלול נגדיר פונקציית עלות $C_t\left(X_t,U_t\right)$מהצורה :

$$C_t\left(X_t,U_t\right)=\left(X_t-X_t^{REF}\right)^TQ_t\left(X_t-X_t^{REF}\right)+\left(U_t-U_t^{REF}\right)^TR_t\left(U_t-U_t^{REF}\right)$$

כאשר מטריצות העלות Q ו-R הנן מטריצות ריבועיות ואלכסוניות, קבועות בזמן, המוגדרות באופן הבא. בחירת הערכים לאחר סבבי אופטימיזציה רבים אשר הניבו את התוצאות המיטביות מבחינת טיב העקיבה וזמן ההתכנסות של הבקר :

$$Q=\begin{pmatrix}10&0&0\\0&10&0\\0&0&1\end{pmatrix}$$

$$R=\begin{pmatrix}0&0\\0&1\end{pmatrix}$$

להלן משוואות העדכון של פתרון בעיית התכנון הדינאמי כפי שהוצג בתרגול בכיתה עבור פונקציית העלות $C_t\left(X_t,U_t\right)$ שהוצגה לעיל (לאחר חישוב טור טיילור מסדר שני של פונקציית העלות, גזירת פונקציית העלות והשוואתה לאפס) :

$$d_t\triangleq-\left(2R_t^T+B_t^TS_{t+1}B_t\right)^{-1}\left(2U_t^TR_t-2U^{REF^T}R_t+B_t^Ts_{t+1}\right)$$

$$K_t\triangleq-\left(2R_t^T+B_t^TS_{t+1}B_t\right)^{-1}B_t^TS_{t+1}A$$

כאשר s_{t+1} מייצג את סכימת אופק הזמן קדימה של הנגזרת הראשונה של פונקציית העלות לפי X :

$$s_{t+1} \triangleq \frac{\partial V}{\partial X}(t+1) = \sum_{i=t+1}^{N} 2\left(X_i^T - X_i^{REF^T}\right) Q_i$$

ולעומת זאת האיבר S_{t+1} מייצג את סכימת אופק הזמן קדימה של הנגזרת השנייה של פונקציית העלות לפי X :

$$S_{t+1} \triangleq \frac{\partial^2 V}{\partial^2 X}(t+1) = \sum_{i=t+1}^{N} 2Q_i$$

במימוש, נעשה שימוש בפועל במטריצה זו בתוספת איבר רגולריזציה מהצורה ρI בכדי לשפר את היציבות הנומרית. ערכי ה-ρ שנבחרו במימוש בפועל מתוארים בהמשך.

המטריצה A_{t}מייצגת את נגזרת הדינמיקה הלא-ליניארית של המערכת ע"פ משתני המצב, כלומר :

$$A_{\mathrm{t}} = \frac{\partial f}{\partial X} = \begin{pmatrix} 0 & 0 & -v sin\theta \\ 0 & 0 & v cos\theta \\ 0 & 0 & 0 \end{pmatrix}$$

המטריצה B_{t}מייצגת את נגזרת הדינמיקה הלא-ליניארית של המערכת ע"פ פקודות הבקרה, כלומר :

$$B_{\mathrm{t}} = \frac{\partial f}{\partial U} = \begin{pmatrix} cos\theta & 0 \\ sin\theta & 0 \\ \frac{tg\delta}{d} & \frac{v}{dcos^2(\delta)} \end{pmatrix}$$

האיברים d_t, K_t מאפשרים לנו לחשב את δU המייצג את השינוי הנדרש ביחס לפקודות הבקרה U באיטרציה הנוכחית בבואנו לחשב את האיטרציה הבאה של הבקר. צורת החישוב בפתרון לאחור ($Backwards\ Pass$) הינו :

$$\delta U_t^* = K_t \delta X_t + \alpha d_t$$

הפרמטר α אמור להיות 1 ע"פ הפיתוח התיאורטי. מאחר והפתרון הנ"ל הינו קירוב ליניארי בלבד, הניסיונות האמפיריים שלנו הוכיחו כי בחירת $\alpha = 1$ גורם לצעדי תיקון אגרסיביים מדי שלא מאפשרים התכנסות נאותה של הבקר. במקום זאת, נבחרו ערכי

α קטנים יותר ומשתנים כתלות בטיב ההתכנסות בדומה לרעיון ה-$Line\ Search$ באופטימיזציית $Gradient\ Descent$. הערכים עצמם והלוגיקה מתוארת בהמשך.

לאחר חישוב δU_t^* נוכל לחשב את הצמד $\left\{X^{\text{i}}, U^i\right\}$ של האיטרציה הבאה (i+1), ע"פ הנוסחאות שהוצגו בתרגול:

$$U_{\text{t}}^{i+1} = U_{\text{t}}^i + K_t\left(X_t^{i+1} - X_t^i\right) + \alpha d_t$$
$$X_{t+1}^{i+1} = X_t^{i+1} + dt \cdot f\left(X_t^{i+1}, U_{\text{t}}^{i+1}\right)$$

כך נוכל לחשב את המסלול באיטרציה הבאה במעבר ה-$Forward\ Pass$.

ר ה-$iLQR$ זקוק לקלט של "מסלול" הרפרנס אליו נרצה להתכנס - מוגדר כצמד $\left\{X^{REF}, U^{REF}\right\}$ וכן מסלול "ניחוש" - $\left\{X^{INIT}, U^{INIT}\right\}$ - איתו נדרש לאתחל את האלגוריתם, בתקווה שהאלגוריתם יתכנס ממנו אל פתרון אופטימאלי הקרוב דיו אל פתרון הרפרנס.

מסלול הרפרנס $\left\{X^{REF}, U^{REF}\right\}$ חושב בקלות בעזרת נוסחאות ה-$Differential\ Flat\ Model$ של דינמיקת רכב אקרמן ע"פ הנוסחאות שהוצגו בתרגולים:

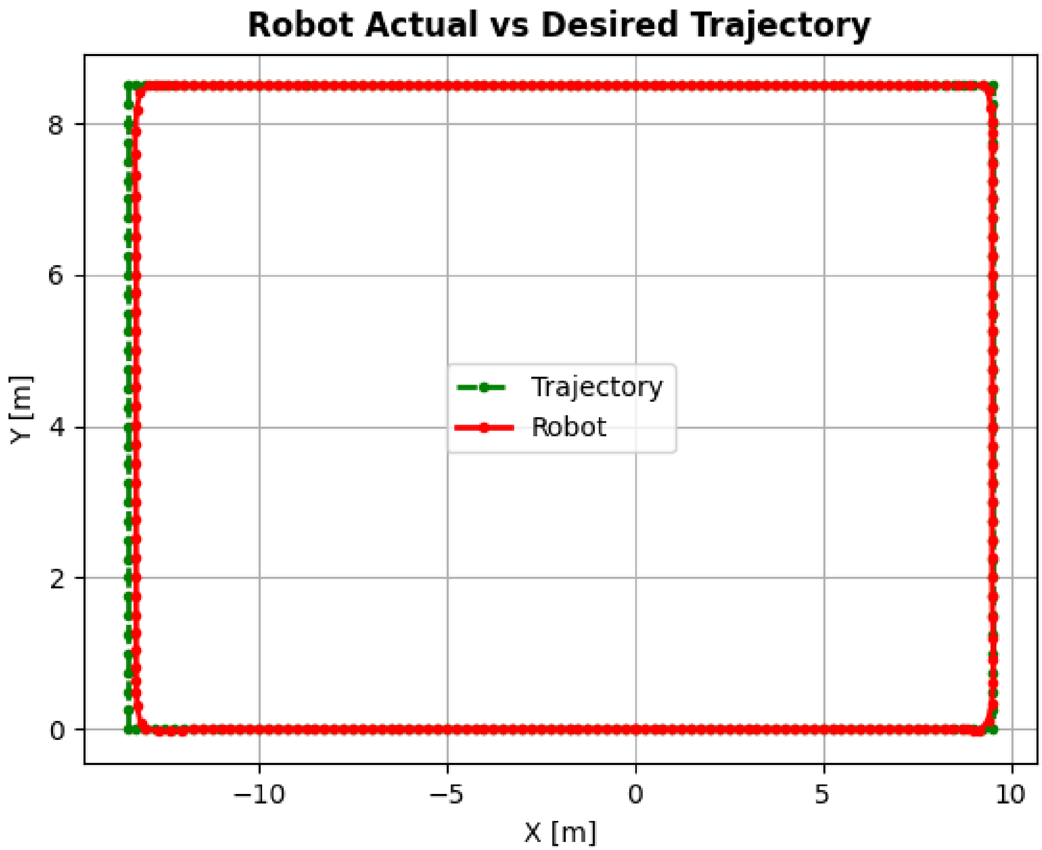
$$X(t) = \begin{pmatrix} x(t) \\ y(t) \\ \theta(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ y(t) \\ Atan2\left(\dot{y}(t), \dot{x}(t)\right) \end{pmatrix}$$

$$U(t) = \begin{pmatrix} v(t) \\ \delta(t) \end{pmatrix} = \begin{pmatrix} \frac{\dot{x}(t)}{cos[Atan2(\dot{y}(t), \dot{x}(t))]} \\ tan^{-1}\left(\frac{d\dot{\theta}(t)}{v(t)}\right) \end{pmatrix}$$

חישוב זה מומש בפונקצייה בשם get_diff_flat_X_and_U המקבלת כקלט את מסלול הרפרנס שסופק בקובץ ref_traj.npy. נשים לב שהמשוואות התיאורטיות הללו אינן לוקחות בחשבון את מגבלת המהירות המרבית של הרכב (0.5 מטר לשנייה) ועל כן ברור שמסלול הרפרנס שיופק הינו אופטימי מדי ולא פיזיבלי הלכה למעשה בבקר שירוץ בסימולציית ה-ROS. עם זאת, זה מסלול רפרנס לגיטימי לשאוף אליו כחלק מהאופטימיזציה.

לעומת זאת, חישוב מסלול "הניחוש" הראשוני הנו מאתגר ולא טריוויאלי. מצד אחד נרצה מסלול שקל להפיקו ומצד שני לא יהיה רחוק מדי ממסלול הרפרנס האידיאלי כדי שאכן הבקר יצליח להתכנס אליו. לטובת כך, הוקלט מסלול - הצמד $\{X, U\}$- עבור בקר ה-$Pure\ Pursuit$. לאחר מכן בוצעה הרצה של סקריפט $Offline$ שאיננו חלק מהגשת התרגיל, בו בקר ה-$iLQR$ רץ באתחול של מסלול בקר ה-$Pure\ Pursuit$ בשאיפה להתכנס למסלול הרפרנס $\left\{X^{REF}, U^{REF}\right\}$. מסלול זה חושב על-פני הקפה מלאה ומאחר והוא נדרש לבנייה רק פעם אחת, הריצה בוצעה ללא מגבלת זמן תוך שימוש בהיפר-פרמטרים "נדיבים" בכדי לאפשר לאלגוריתם להתכנס בצורה טובה לפתרון מסלול "איכותי". לאחר מספרי אלפי ריצות התקבל פתרון שאכן מתחשב במגבלות המהירות של הרכב והוא מהווה "מסלול" ניחוש איכותי להתחיל ממנו בכל נקודת זמן בו נחפוץ. המסלול נשמר בשני הקבצים הבאים: u_initial.npy, u_initial.npy. יש לטעון קבצים אלו ל-$Docker$ לתיקיית $resources$ (לאותה תיקייה בה נמצא קובץ ref_traj.npy המסלול ref_traj.npy).

הדיאגרמות הבאות מציגות את מסלול הניחוש $\{X^{INIT}, U^{INIT}\}$ שהתקבל ובו הבקר ישתמש בזמן אמת:



כפי שניתן לראות התקבלה התכנסות טובה מאוד של הבקר (מסלול הרובוט הצפוי באדום) אל מסלול הרפרנס שהופק ע"י מודל האקרמן ע"י ה-$Differential\ Flat\ Model$ (קו ירוק).

נקודות ודילמות מרכזיות באופן המימוש של הבקר:

1. אופק הבקרה

כזכור תדר הסימולציה הינו חצי שנייה. כלומר, הבקר חייב לסיים את חישוביו תוך פחות מחצי שנייה. נכתבה סביבת $Offline$ בה פותח הקוד וכן נעשו ניסיונות רבים (מאוד) למציאת ההיפר-הפרמטרים המיטביים שיאפשרו הן התכנסות של המסלולים והן יעמדו מבחינת זמן המחזור של מחצית השנייה של הסימולציה. בכדי לעמוד באילוץ קשה זה, נמצא כי בזמן אמת של הריצה, לא ניתן לתכנן מסלול איכותי ליותר מ-3 נקודות קדימה. כלומר, בכל נקודת זמן, הבקר טוען ממסלול הרפרנס $\{X^{REF}, U^{REF}\}$ וממסלול הניחוש $\{X^{INIT}, U^{INIT}\}$ את 3 איברי המסלול מהזמן הנוכחי וקדימה. המצב המדוד של משתני המצב X בזמן המחזור הנוכחי משמש כאילוץ לאיבר הראשון של X^{INIT}.

2. עדכון היפר-פרמטרים מאיטרציה לאיטרציה

מומשה לוגיקה הבוחנת את השינוי בפונקציית העלות מאיטרציה לאיטרציה ומעדכנת את ההיפר-פרמטרים הבאים של האלגוריתם:

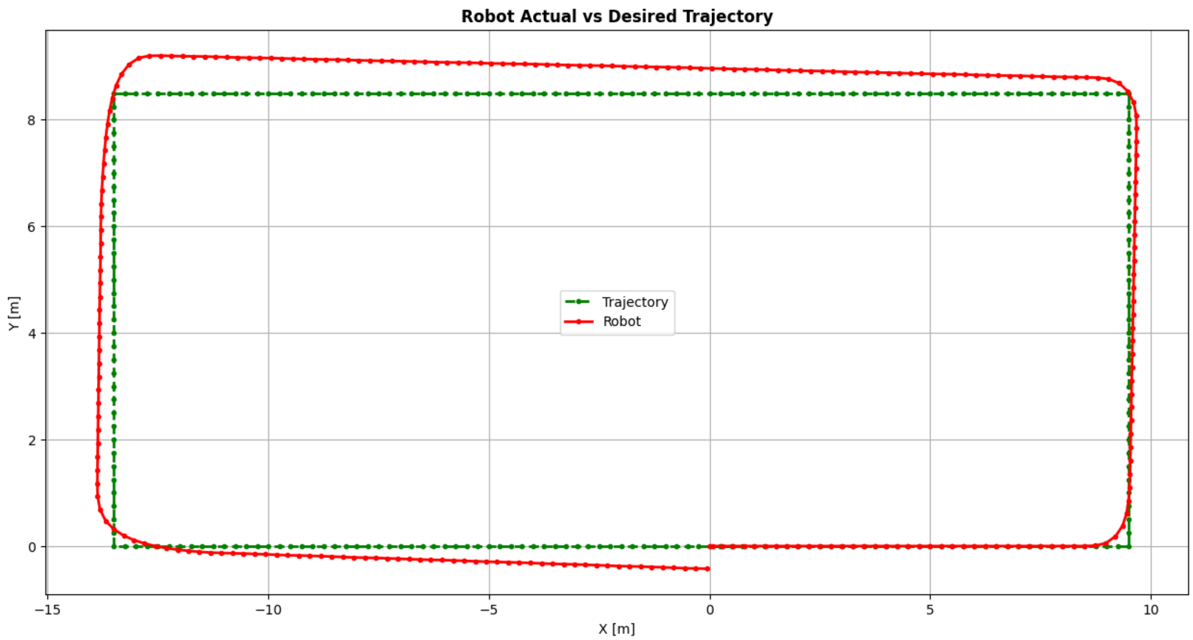
ρ השולט בתוספת הרגולריזציה לאיבר S_{t+1} וכן הפרמטר α השולט בגודל צעד התיקון בין האיטרציות המכפיל את האיבר d_t.

הערך ρ נבחר להיות 10 אשר פוחת באופן מעריכי במידה וחל שיפור בערך פונקציית העלות. הפרמטר α נבחר לערך ראשוני של 0.2 וגם הוא מעודכן ע"פ השינוי בפונקציית העלות. התכנסות הוגדרה כאשר השיפור היחסי של תוצאת ערך העלות החדש אינו טוב מ-10^{-6} ביחס לערך העלות מהאיטרציה הקודמת.

כוונון כל הפרמטרים הנ"ל בוצע בסביבת $Offline$ ייעודית המאפשרת ריצה נוחה וקלה לדיבוג של הקוד ללא תלות בסימולציית ה-ROS.

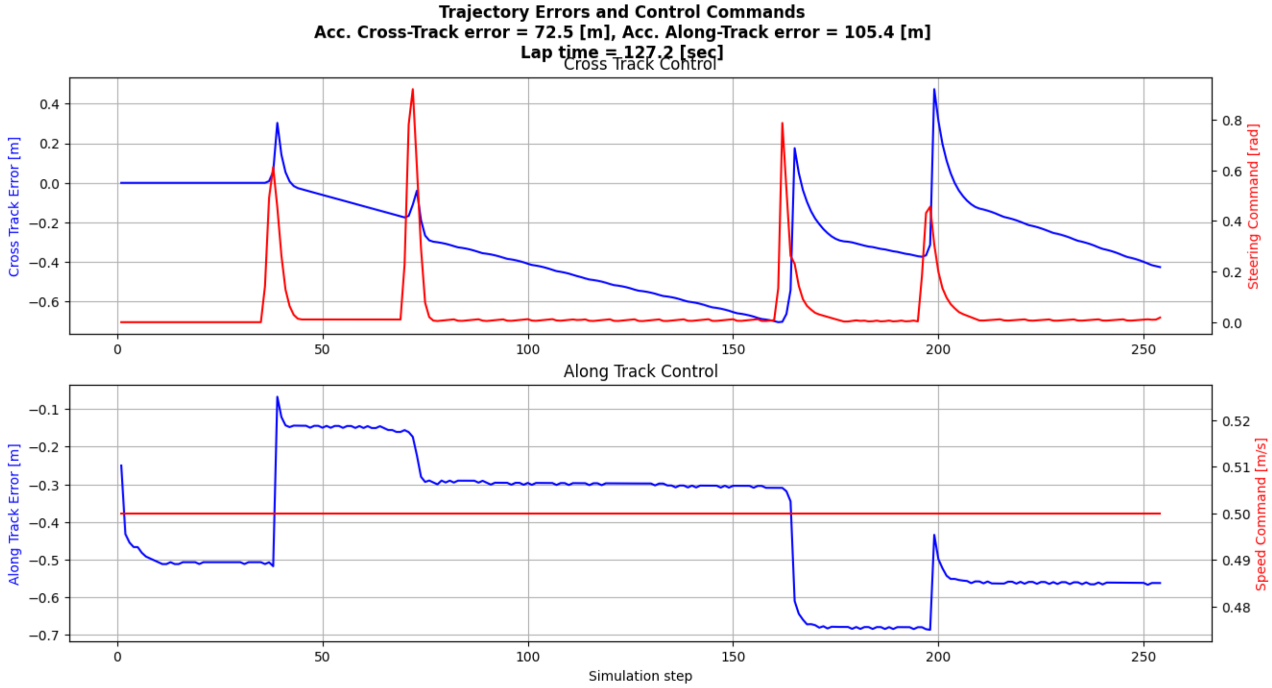
<u>תוצאות ריצת הבקר בסימולציה</u>

הבקר שפותח הורץ כחלק מסימולציית ה-ROS. להלן דיאגרמת עקיבת המסלול שהתקבלה בריצת זמן אמת :



כפי שניתן לראות בחלק מהסלול מתקיימת עקיבה טובה בעוד בחלקים אחרים ניכר כי מסלול הרובוט אינו בעקיבה מספיק טובה. ללא ספק ניכר כי נדרשות איטרציות נוספות לבקר בכדי להתכנס (עם זאת הוא בהחלט בכיוון). מפאת הקושי לאזן הן את העמידה בזמן המחזור של הסימולציה ובמקביל לשמר איכות עקיבה סבירה של פתרונות הבקר, לא הצלחנו לכוון את הבקר לכיוון טוב יותר וזהו הכוונון המיטבי מבחינתנו.

מבחינת שגיאות הבקרה התקבלו הביצועים הבאים :



כפי שניתן לראות, בקרת ה $Cross\,Track$ נסחפת עם הזמן. למרות ניסיונות רבים למצוא היפר-פרמטרים טובים יותר - זהו הכוונון הטוב ביותר שהצלחנו לייצב אשר מאפשר הן עמידה בזמן המחזור והן התכנסות מסלולים המאפשרת לרכב להשלים הקפה שלמה בצורה תקינה.

סעיף 3

להלן טבלה המרכזת השוואה בין 3 סוגי הבקרים:

Controller	Lap Time [sec]	Total e_{at} $[m]$	Total e_{ct} $[m]$	Notes
PID	127.4	175.9	15.4	$ct: K_p, K_i, K_d = (1,0,2)$ $at: K_p, K_i, K_d = (5,0,0)$
Pure Pursuit	126.0	20.2	4.8	Lookahead Distance 3
iLQR	.2127	105.4	72.5	

שאלה 4

בבקר האופטימלי שילבנו רכיב PID שיבקר על המהירות (תוך הסרת המגבלה המלאכותית על המהירות) ורכיב PP שיבקר על הזווית.

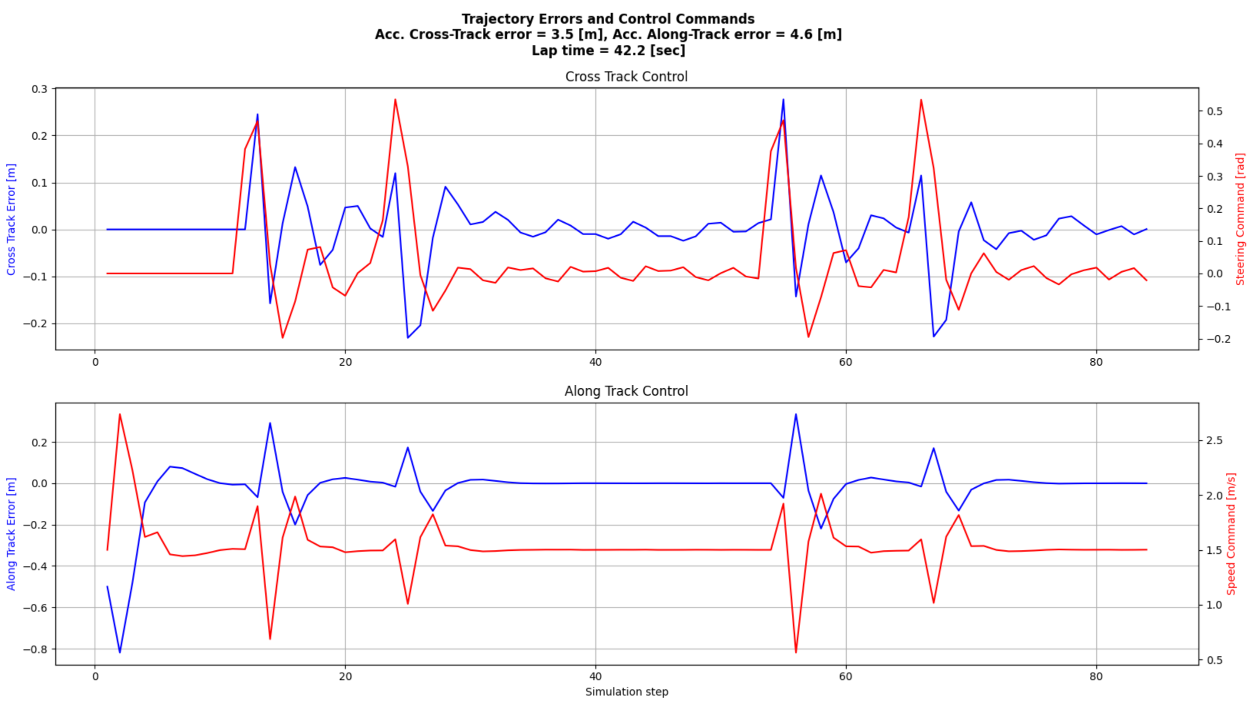
בנוסף האצנו את המסלול ע"י הוספת תנאי בפונקציה get_ref_pose כך שאם אנחנו משתמשים בבקר אופטימלי - מקדמים את $self.waypoint\ index$ בצעד נוסף.

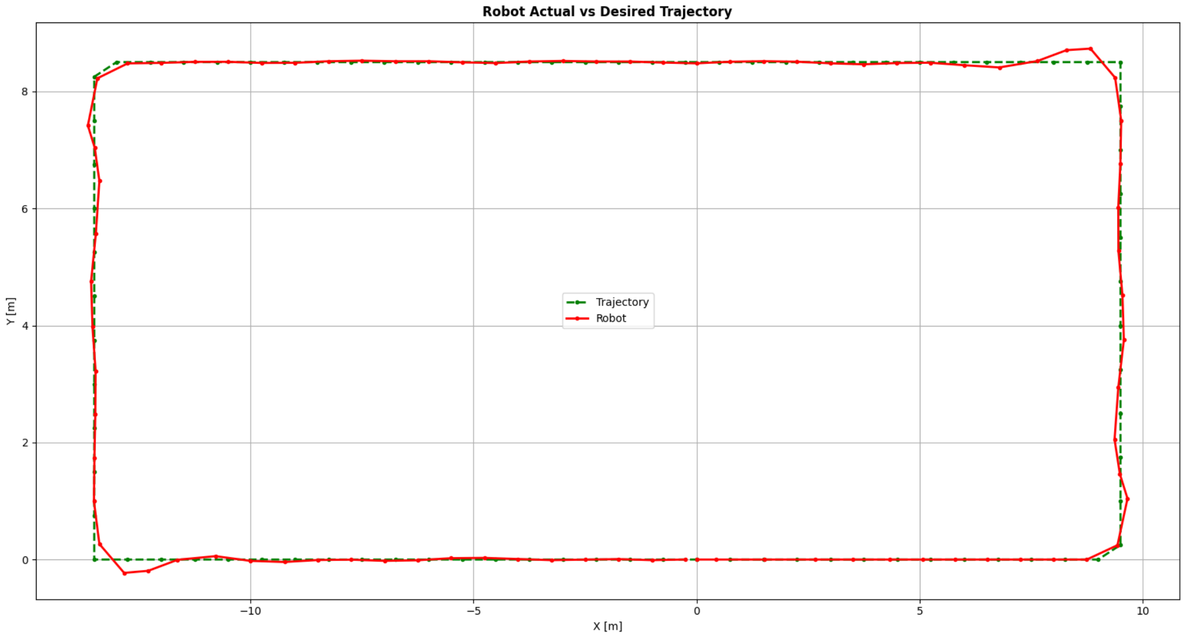
הדבר יצר אוסצילציות חזקות מאוד כאשר השתמשנו ב$lookahead = 3$, מה שגרם לנו לבצע סדרת ניסויים מחודשת עם ערכי $lookahead$ שונים. התוצאות להלן:

Lookahead Distance	Lap Time [sec]	Total e_{at} $[m]$	Total e_{ct} $[m]$	Notes
8	.16342	.1712	4.93	
7				Not Tested
6	42.162	9.08	.053	
5	.16642	.657	2.71	
4	42.17	7.05	3.21	
3	5.1642	.317	12.87	

זיהינו שבאופן עקבי לאחר פניות הרכב עוצר לרגע ו"מתאושש". על מנת לטפל בסוגיה שיחקנו עם ערכי הPID, ובפרט הגדלנו את האינטגרטור על מנת לגרום לרכב להפסיק לעצור. הרכב לא מפסיק להאט, אבל כתופעת לוואי קיבלנו שיפור בביצועי הבקר.

להלן ביצועי הבקר הסופי:




Robot Actual vs Desired Trajectory
Trajectory
Robot
Y [m]
X [m]
0
2
4
6
8
−10
−5
0
5
10