به نام خداوند

تمرین شماره ۶ رباتیک

عرفان اعتصامی ۹۶۱۰۶۲۱۴

مقدمه) در حل این تمرین از مطالب موجود در جزوه‌ی جلسات ۱۷ و ۱۸ استفاده شده است. برنامه‌ی $Run.m$ مسئول اجرای برنامه می‌باشد. همچنین تابع $Path\_generator.m$ مسئول تولید مسیر با روش میدان پتانسیل بوده و تابع $Random\_walk.m$ وظیفه‌ی پیاده‌سازی الگوریتم $random\ walk$ را بر عهده دارد. لازم به ذکر است که چون در این تمرین تنها نیاز به طراحی هندسه‌ی مسیر می‌باشد، واحد طول مخصوصی برای موقعیت نقاط در نظر گرفته نشده است.

روش میدان پتانسیل) برای پیاده‌سازی روش میدان پتانسیل از الگوریتم موجود در جزوه‌ی جلسه‌ی ۱۸ استفاده شده است.

$K \leftarrow 1$

$\vec{x}(K) \leftarrow \vec{x}_s$

while $|\vec{x}(K) - \vec{x}_E|$ is "Large"

$\vec{F}_{att} \leftarrow -\eta(\vec{x}(K) - \vec{x}_E)$

for $i = 1$ to m — m تعداد موانع

Find $\vec{B}_i$

$\rho_i \leftarrow |\vec{B}_i - \vec{x}(K)|$

if $\rho_i \leq \rho_{0i}$

شکل ۱) الگوریتم روش میدان پتانسیل - الف

$$\vec{F}_{rep,i} \leftarrow \alpha_i \frac{1}{\rho_i^3}\left(\frac{1}{r_i} - \frac{1}{\rho_{0i}}\right)(\vec{x}(k) - \vec{B}_i)$$

else

$$F_{rep_i} \leftarrow 0$$

end for

$$\vec{F} \leftarrow \vec{F}_{att} + \sum_{i=1}^{m} F_{rep_i}$$

$$\vec{F}_{drive} = \frac{\vec{F}}{|\vec{F}|}$$

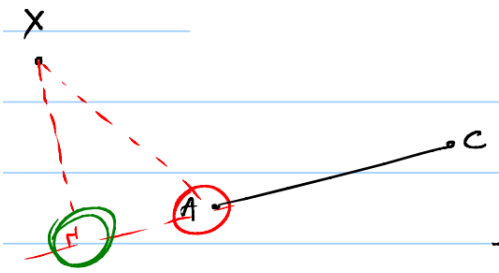
$$\vec{x}(k+1) \leftarrow \vec{x}(k) + \varepsilon \vec{F}_{drive}$$

$$k \leftarrow k+1$$

end while

شکل ۲) الگوریتم روش میدان پتانسیل - ب

برای پیدا کردن نزدیک‌ترین نقطه‌ی مسیر به موانع ($B_i(\vec{x})$) نیز از نتیجه‌ی موجود در جزوه‌ی همین جلسه استفاده می‌کنیم. همان‌گونه که در ادامه نیز اشاره خواهد شد، در پیاده‌سازی الگوریتم $random\ walk$ برای تشخیص عدم برخورد به مانع ضمن حدس زدن اتفاقی طول و زاویه‌ی پرش، مشابه با روشی که در شکل‌های ۳ تا ۵ نمایش داده شده است، عمل کرده‌ایم. لازم به ذکر است که موانع در برنامه‌ی نوشته شده به این گونه باید تعریف شوند که نقطه‌ی اول آن‌ها دارای مختصه‌ی x کمتری باشد و در صورت برابری مختصه‌ی x هر دو، نقطه‌ی اول باید مختصه‌ی y کمتری داشته باشد.



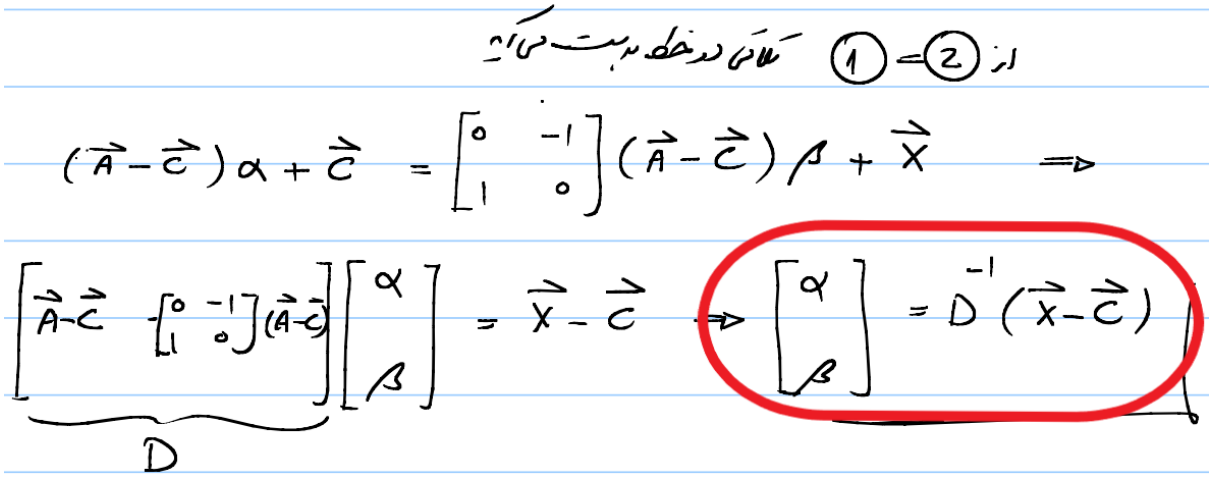
معادله خطی که از C, A می‌گذرد:

$$\vec{P} = (\vec{A} - \vec{C})\alpha + \vec{C} \qquad \alpha \in \mathbb{R} \quad (1)$$

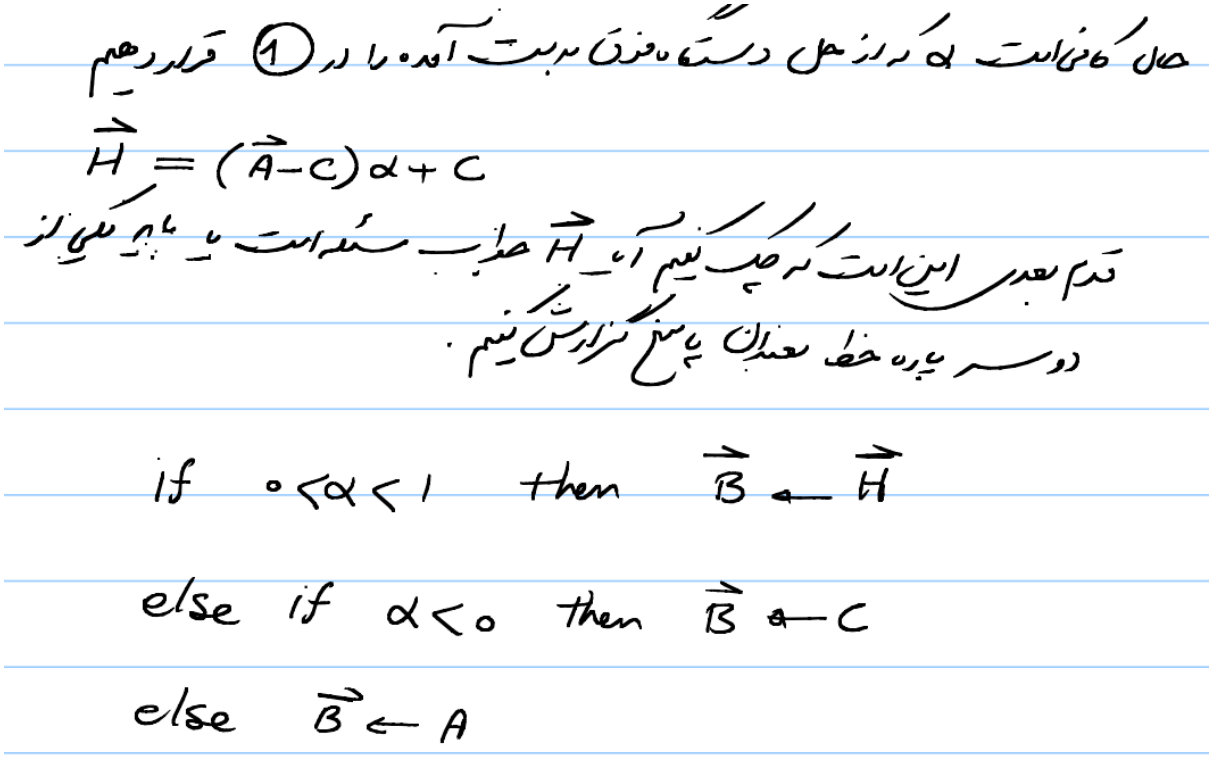
معادله خطی که از X می‌گذرد و عمود بر AC باشد

$$\vec{q} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}(\vec{A} - \vec{C})\beta + \vec{X} \qquad \beta \in \mathbb{R} \quad (2)$$

شکل ۳) پیدا کردن نزدیک ترین نقطه‌ی مسیر به موانع - الف

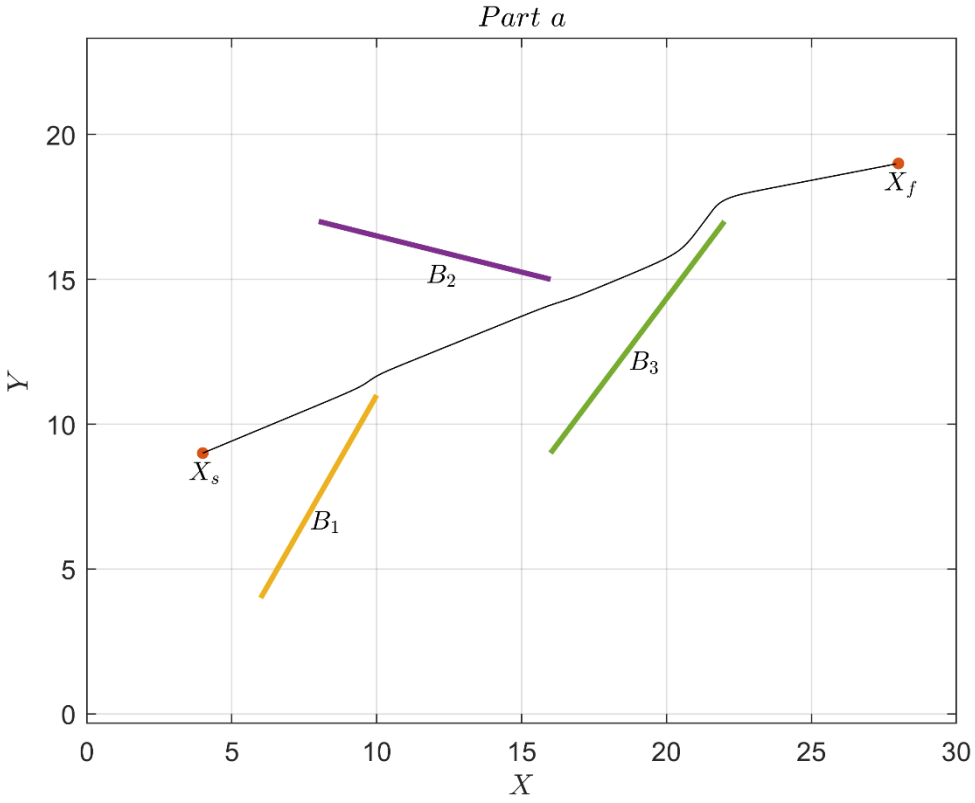


شکل ۴) پیدا کردن نزدیک ترین نقطه‌ی مسیر به موانع - ب



شکل ۵) پیدا کردن نزدیک ترین نقطه‌ی مسیر به موانع - ج

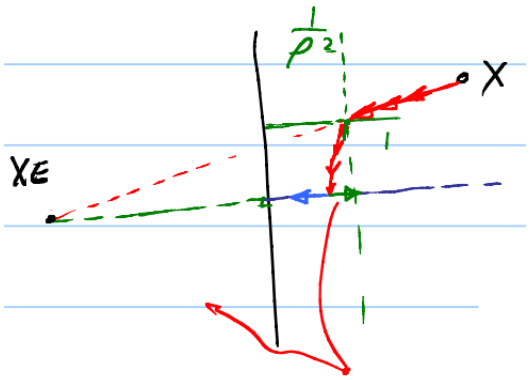
پرسش الف) با پیاده‌سازی الگوریتم‌های توضیح داده شده در تابع $Path\_generator.m$، مسیر بین $\vec{X}_S$ و $\vec{X}_F$ با روش میدان پتانسیل تولید شده است (متغیر $Path\_a$). نتیجه در نمودار ۱ مشاهده می‌شود. لازم به ذکر است که علاوه بر ورودهای اشاره شده در صورت پرسش برای این تابع، یک ورودی $flag$ نیز به صورت اضافه در نظر گرفته شده است که اگر ۰ باشد، الگوریتم $random\ walk$ اجرا نمی‌شود و اگر ۱ باشد، تابع تولید مسیر همراه با فراخوانی تابع $Random\_walk.m$ اجرا می‌شود.



نمودار ۱) تولید مسیر – پرسش الف

همان‌گونه که مشاهده می‌شود، قسمت عمده‌ی این مسیر به صورت حرکت بر روی یک خط راست می‌باشد که هنگام نزدیک شدن به موانع و فاصله‌ی تاثیر آن‌ها، تغییر جهت می‌دهد.

مشکل کمینه‌ی محلی) یکی از مزیت‌های ویژه‌ی روش میدان پتانسیل، بدون $iteration$ بودن آن است که موجب می‌شود بتوان حجم زیادی از محاسبات را در زمان به کمک آن انجام داد. با این حال، یکی از ایرادهای این روش، مشکل به دام افتادن آن ضمن رسیدن به کمینه‌ی محلی می‌باشد. مطابق توضیحات انتهای جزوه‌ی جلسه‌ی ۱۸، کمینه‌ی محلی هنگامی رخ می‌دهد که $\vec{F}_{att}$ و $\vec{F}_{rep}$ در یک راستا (و طبیعتا در جهت عکس یک‌دیگر) قرار بگیرند (یا به عبارت دیگر، مانع بر مسیر عمود باشد). در این حالت مجموع این دو بردار نیز روی همان راستا قرار خواهد گرفت و در نتیجه، نقطه‌ی $\vec{X}$ از مسیر نمی‌تواند طبق این الگوریتم از این امتداد خارج شود.



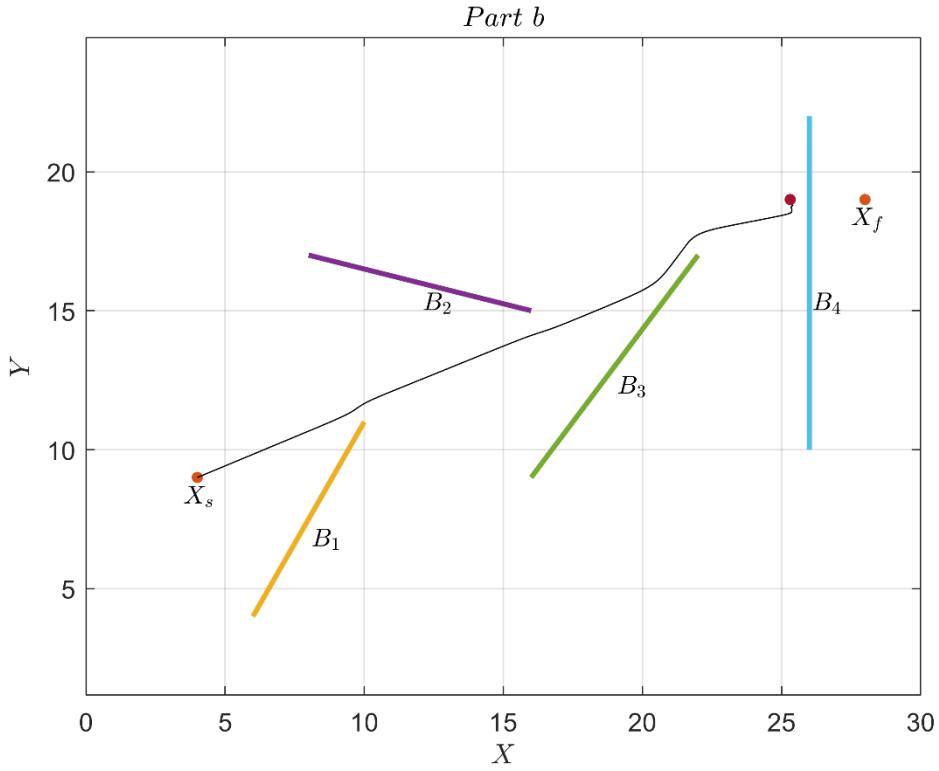
شکل ۶) مشکل کمینه‌ی محلی

پرسش ب) برای این که الگوریتم تولید مسیر را مجهز به تشخیص کمینه‌ی محلی کنیم، شرط این که $\vec{F}_{att}$ و $\vec{F}_{rep}$ در یک راستا و در جهت مختلف یک‌دیگر قرار بگیرند را به تابع $Path\_generator.m$ می‌افزاییم. برای این کار کافی است زاویه‌ی بین این دو بردار برابر با ۱۸۰ درجه شود یا به عبارت دیگر با استفاده از ضرب داخلی، کسینوس زاویه‌ی میان این دو بردار برابر با ۱- شود.

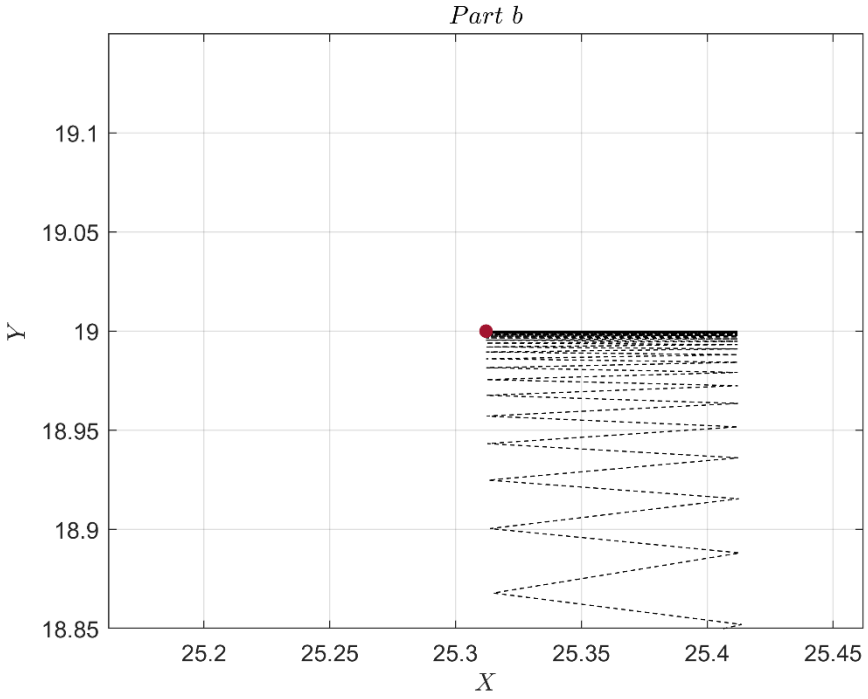
```
% find local minima
direction = dot(Fatt, sum_Frep) / (norm(Fatt) * norm(sum_Frep));
% specific condition
cond = 0;
Threshold2 = 0.05;
if j >= 4
    cond = abs(P(j, 1)-P(j-1, 1))<Threshold2 &&...
           abs(P(j, 1)-P(j-2, 1))<Threshold2 &&...
           abs(P(j, 1)-P(j-3, 1))<Threshold2;
end
%   if direction == -1 || cond
if round(direction, 2) == -1 || cond
```

شکل ۷) تشخیص کمینه‌ی محلی

این کار را به کمک کد فوق انجام داده‌ایم. لازم به ذکر است که با توجه به دقت پیش‌فرض ۱۵ رقم اعشار نرم‌افزار $MATLAB$ در نوع داده‌ی عددی، می‌توان از شرط برابری تقریبی کسینوس زاویه‌ی میان دو بردار با عدد ۱- (به جای برابری دقیق) استفاده کرد تا به این گونه تا حدی از ایجاد اعداد تکراری در مسیر که بعضاً تا ۸ رقم اعشار نیز با یک‌دیگر یکسان هستند جلوگیری نمود. همچنین از آن‌جا که با وجود مانعی که منجر به ایجاد کمینه‌ی محلی می‌شود، ربات در مسیری به موازات مانع و در جهت عمود بر آن شروع به نوسان می‌کند، شرط خاص $cond$ نیز برای یافتن دقیق‌تر کمینه‌ی محلی در نظر گرفته شده است. از آن‌جا که در این تمرین، مانع فقط به صورت عمودی می‌باشد، تنها نوسانات در راستای x لحاظ شده‌اند. نتیجه (متغیر $Path\_b$) با اعمال هر دو شرط و بزرگنمایی در ناحیه‌ی کمینه‌ی محلی در صفحات بعد نمایش داده شده است.

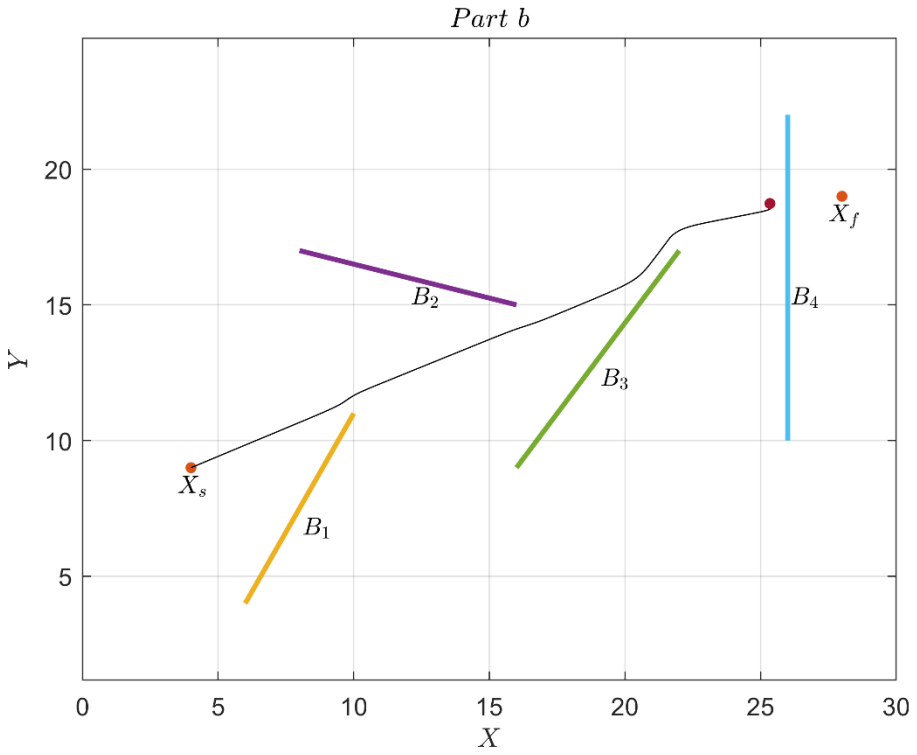


نمودار ۲) تولید مسیر با شرط برابری دقیق کسینوس زاویه‌ی میان دو بردار با ۱- – پرسش ب

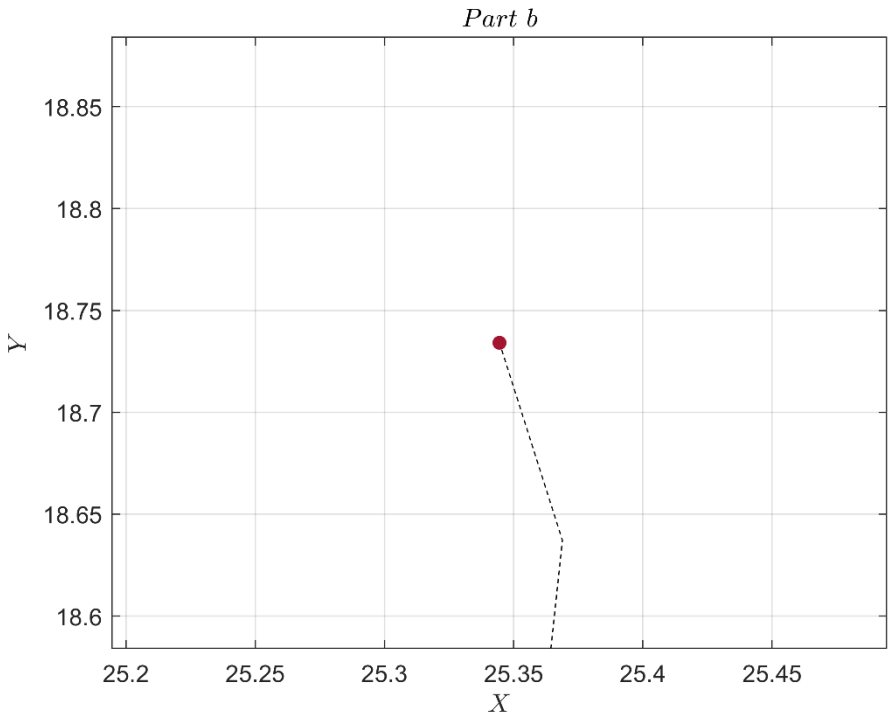


نمودار ۳) بزرگنمایی بر روی کمینه‌ی محلی با شرط برابری دقیق کسینوس زاویه‌ی میان دو بردار با ۱- – پرسش ب

مکان دقیق کمینه‌ی محلی با شرط برابری دقیق کسینوس زاویه‌ی میان دو بردار با ۱- برابر با $\begin{pmatrix} 25.3121 \\ 19.0000 \end{pmatrix}$ می‌باشد.



نمودار ۴) تولید مسیر با شرط برابری تقریبی کسینوس زاویه‌ی میان دو بردار با ۱- – پرسش ب



نمودار ۵) بزرگنمایی بر روی کمینه‌ی محلی با شرط برابری تقریبی کسینوس زاویه‌ی میان دو بردار با ۱- – پرسش ب

مکان تقریبی کمینه‌ی محلی با شرط برابری تقریبی کسینوس زاویه‌ی میان دو بردار با ۱- برابر با $\begin{pmatrix} 25.3445 \\ 18.7341 \end{pmatrix}$ می‌باشد.

در ادامه با توجه به پایدارتر بودن شرط برابری تقریبی، از آن برای حل پرسش ج استفاده می‌کنیم.

الگوریتم $random\ walk$ همان‌گونه که در صورت پرسش توضیح داده شده است، یک روش برای خارج شدن از ناحیه‌ی کمینه‌ی محلی، استفاده از الگوریتم $random\ walk$ می‌باشد. اساس کار به این گونه است که با اجرا شدن این الگوریتم، یک پرش به طول و زاویه‌ی دلخواه در موقعیت فعلی ربات ایجاد می‌شود. بدیهی است این پرش نباید به گونه‌ای باشد که ربات ضمن این پرش به موانع برخورد کند.

همان گونه که پیش‌تر اشاره شد، برای یافتن نزدیک‌ترین نقطه‌ی مسیر به موانع ($B_i(\vec{x})$) مشابه جزوه‌ی جلسه‌ی ۱۸ عمل می‌کنیم. اگر A و C به ترتیب مختصات سر اول و دوم یک مانع و همچنین S و T مختصات موقعیت قبل و پس از پرش باشند، با نوشتن معادله‌ی خطوط به صورت پارامتری خواهیم داشت:

معادله‌ی خط واصل بین A و C: $(A-C)\alpha + C$

معادله‌ی خط واصل بین S و T: $(S-T)\beta + T$

با برابر قرار دادن معادله‌ی دو خط، خواهیم داشت:

$$(A-C)\alpha + C = (S-T)\beta + T \rightarrow [(A-C) \quad -(S-T)]\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = (T-C) \rightarrow$$

$$\begin{bmatrix}\alpha\\ \beta\end{bmatrix} = [(A-C) \quad -(S-T)]^{-1}(T-C)$$

با به دست آمدن مقدار α کافی است شرط زیر را بررسی کنیم.

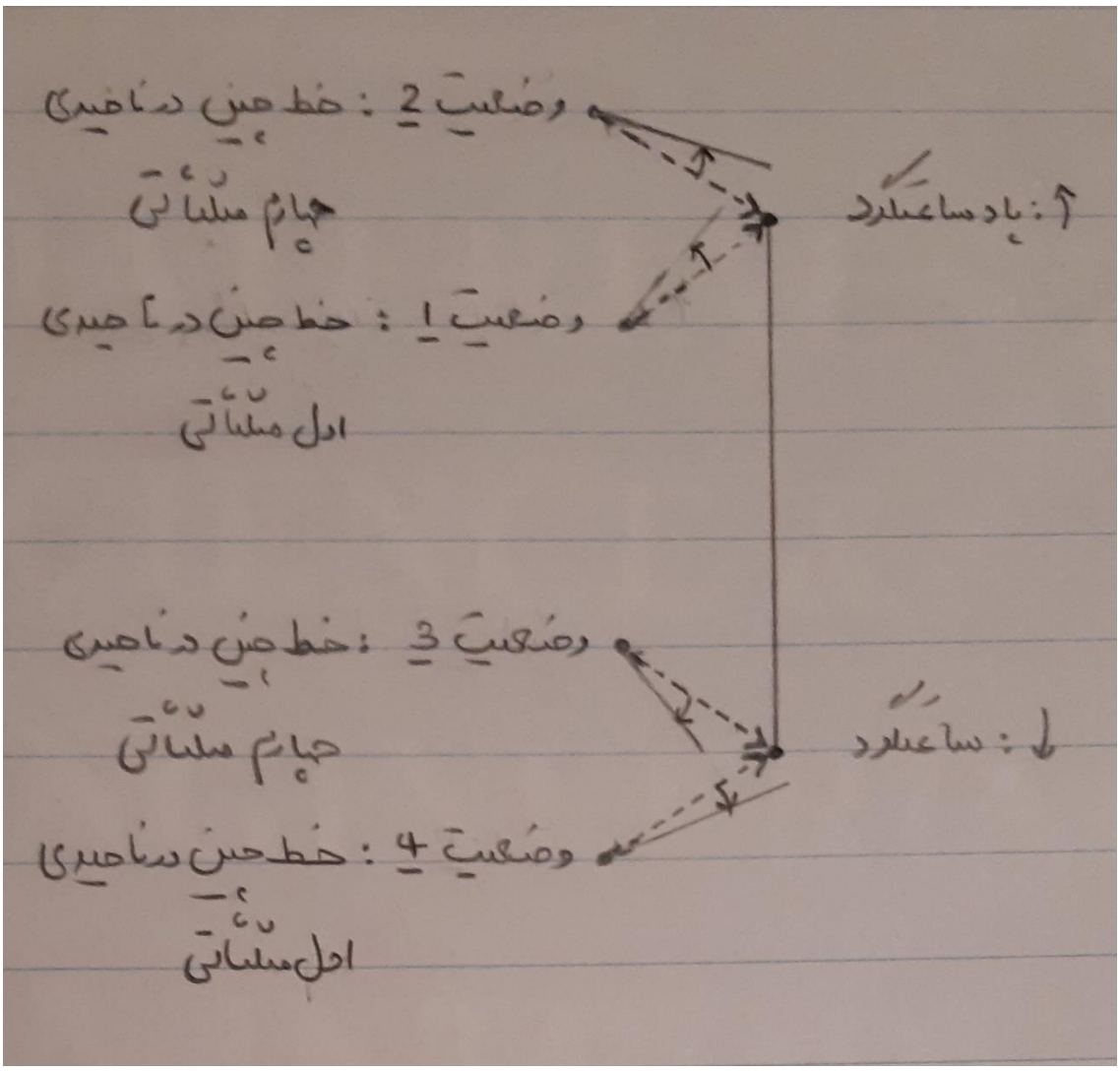
$if\ \alpha \geq 0\ \&\ a \leq 1 \rightarrow$ برخورد مسیر پرش با مانع

حال اگر مسیر پرش با هیچ‌کدام از موانع برخورد نداشته باشد، یعنی این تابع توانسته یک پرش مناسب را تولید کند که مختصات حاصل از آن در مسیر ذخیره شده و روش میدان پتانسیل با مختصات جدید ادامه می‌یابد. توجه شود که حلقه‌ی $while$ موجود در تابع $Random\_walk$، آنقدر تکرار می‌شود تا تابع بتواند یک پرش مناسب تولید کند که شرط عدم برخورد با موانع را ارضاء کند.

منتها پیش از انجام این عملیات لازم است تا بتوانیم یک پرش تا حد ممکن هوشمند (به جای پرش دلخواه) تولید کنیم. منظور از هوشمند آن است که طول پرش و زاویه‌ی آن تا حد ممکن بهینه باشند تا بتوانند ربات را قادر سازند که در صورت امکان تنها با یک پرش، از ناحیه‌ی کمینه‌ی محلی خارج شود. برای این کار در ابتدا مطابق خطوط ۷ تا ۲۳ تابع $Random\_walk.m$، نزدیک‌ترین رأس موانع به موقعیت فعلی ربات و فاصله‌ی بین این دو را می‌یابیم. سپس مطابق خطوط ۲۵ و ۲۶ تابع، با استفاده از مفهوم شیب خط، زاویه‌ی خط واصل از موقعیت فعلی ربات و نزدیک‌ترین رأس شناسایی شده ($angle\_seed$) را به دست می‌آوریم.

حال کمینه فاصله‌ی موقعیت فعلی ربات و نزدیک‌ترین رأس شناسایی شده (min_dist) را برای احتیاط با مقدار 0.1 جمع می‌کنیم و مطابق با خطوط ۳۶ تا ۴۰ تابع، طول پرش هوشمند را انتخاب می‌کنیم.

در مرحله‌ی بعد، بسته به این که ربات در چه وضعیتی به مانعی که باعث ایجاد کمینه‌ی محلی شده نزدیک شده باشد، مطابق شکل ۸، چهار حالت پیش می‌آید؛ علی‌رغم این که مانع به صورت عمودی رسم شده است، این چهار حالت، آن گونه که در برنامه‌ی تابع تعریف شده‌اند، در هر وضعیت دیگری نیز قابل اعمال هستند. فلش‌های موجود در این شکل، بیانگر زاویه‌ی پیشنهادی پرش برای خروج از ناحیه‌ی کمینه‌ی محلی و عدم برخورد به مانع هستند. این مقدار در قالب ثابت $cst1 = \frac{\pi}{36} rad = 5°$ تعریف شده است. همچنین برای این که اندکی فاصله میان خط راست واصل میان موقعیت فعلی و موقعیت پس از پرش با مانع وجود داشته باشد، ثابت $cst2 = 2 \times \frac{\pi}{180} rad = 2°$ به این مقدار مطابق برنامه‌ی تابع افزوده می‌شود. حال مقدار زاویه‌ی پرش با توجه به وضعیت ربات نسبت مانع، در دو بازه‌ی $angle\_seed + (2°, 7°)$ (ربع اول مثلثاتی) یا $angle\_seed - (2°, 7°)$ (ربع چهارم مثلثاتی) انتخاب می‌شود.

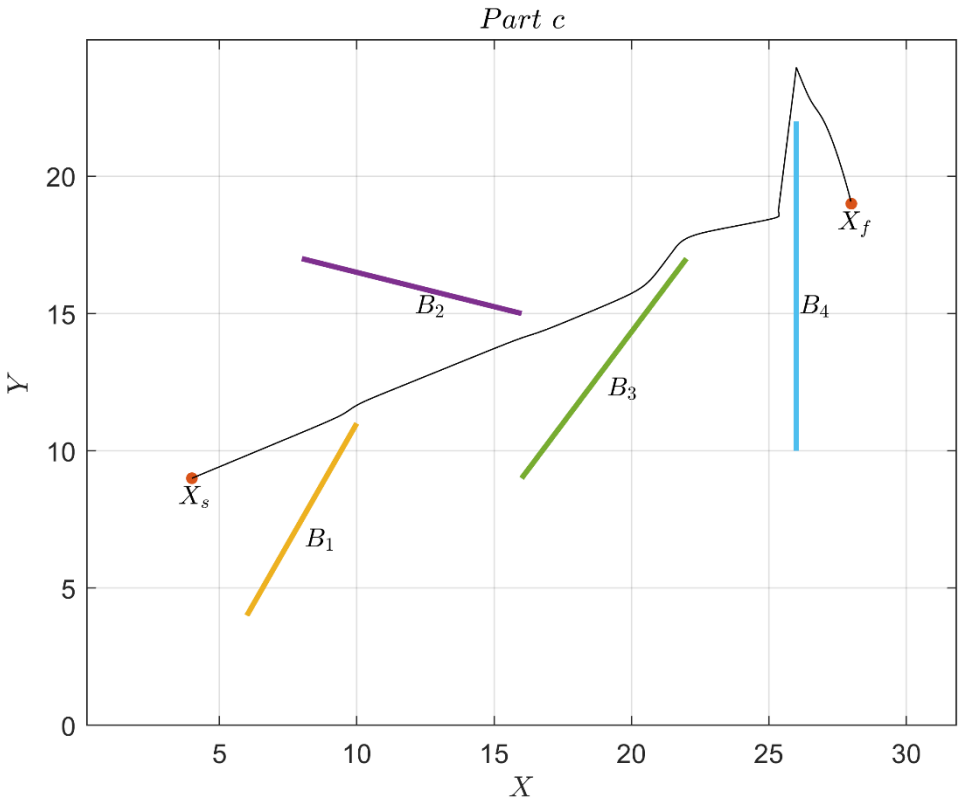


شکل ۸) تعریف زاویه‌ی پرش هوشمند

پرسش ج) با توجه به این که اندازه و زاویه‌ی پرش تولید شده چگونه باشد، مسیرهای بی‌شماری تولید می‌شود که ربات با یک پرش از ناحیه‌ی کمینه‌ی محلی خارج می‌شود. لازم به ذکر است که در صورت ازدیاد تعداد پرش‌ها (متوقف شدن نسبی در روش میدان پتانسیل) می‌توان بازه‌ی تغییرات زاویه‌ی پرش را محدودتر (کاهش $cst1$) یا کمینه‌ی طول ممکن پرش (min_dist) را تغییر داد. در ادامه چند حالت عمومی و خاص قرارگیری مانع چهارم بررسی شده است.

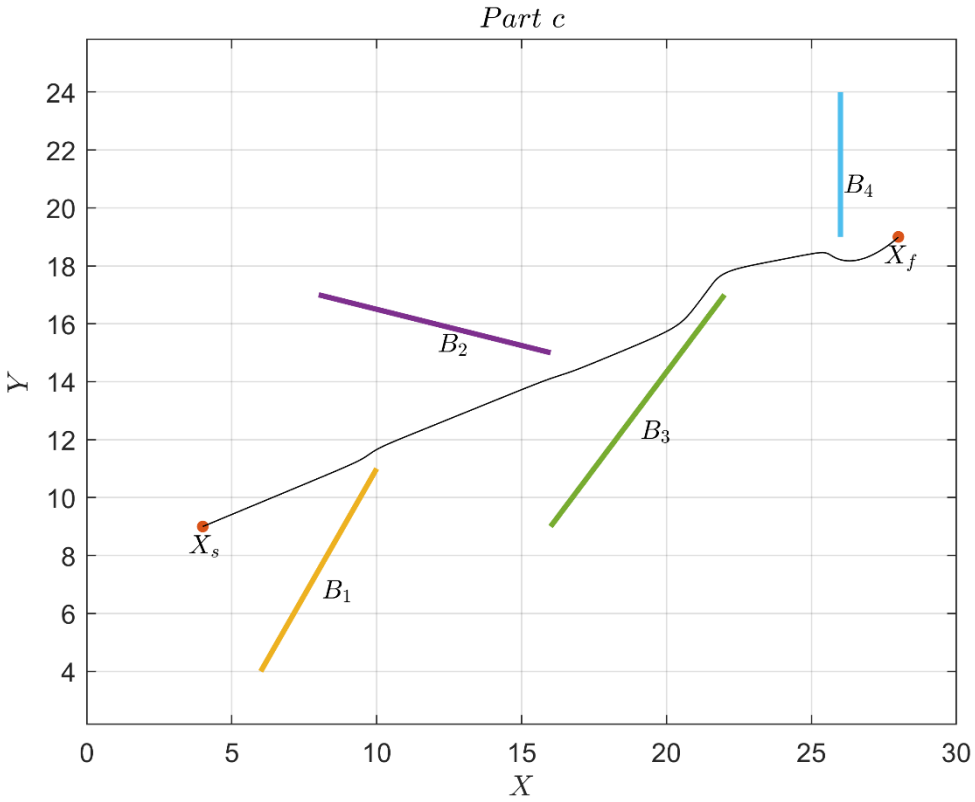
لازم به ذکر است که در تمام حالاتی که در ادامه بیاید، امکان دارد ربات از $\vec{X}_f$ عبور کند (شرایط خاص ۱) در این صورت برنامه به پایان می‌رسد. با انتخاب مقادیر متفاوت برای مانع چهارم، به نظر می‌رسد که عبور از $\vec{X}_f$ علاوه بر نحوه‌ی پیاده‌سازی الگوریتم $random\ walk$، به مقدار ϵ در روش میدان پتانسیل نیز مرتبط است و برای حل این مشکل، ϵ نیز باید تغییر کند؛ در غیر این صورت، برنامه در حلقه‌ی بی‌نهایت به دام می‌افتد. همچنین به دلیل نوع تعریف شناسایی کمینه‌ی محلی و دقت محاسبات عددی، ممکن است زوائدی در نزدیکی کمینه‌ی محلی مشاهده شود.

۱) اگر مختصات دو رأس مانع چهارم به صورت $\binom{26}{10}$ و $\binom{26}{22}$ باشد:



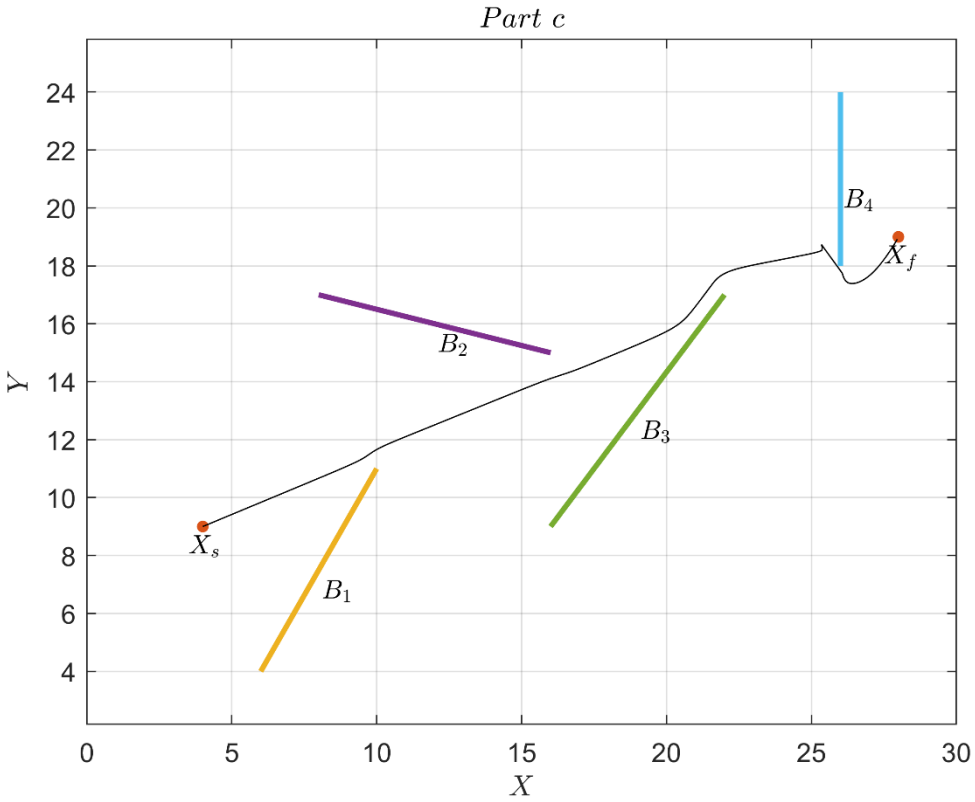
نمودار ۶) یک مسیر ممکن با قرارگیری مانع چهارم در وضعیت ۱ – پرسش ج

۲) اگر مختصات دو رأس مانع چهارم به صورت $\binom{26}{19}$ و $\binom{26}{24}$ باشد:



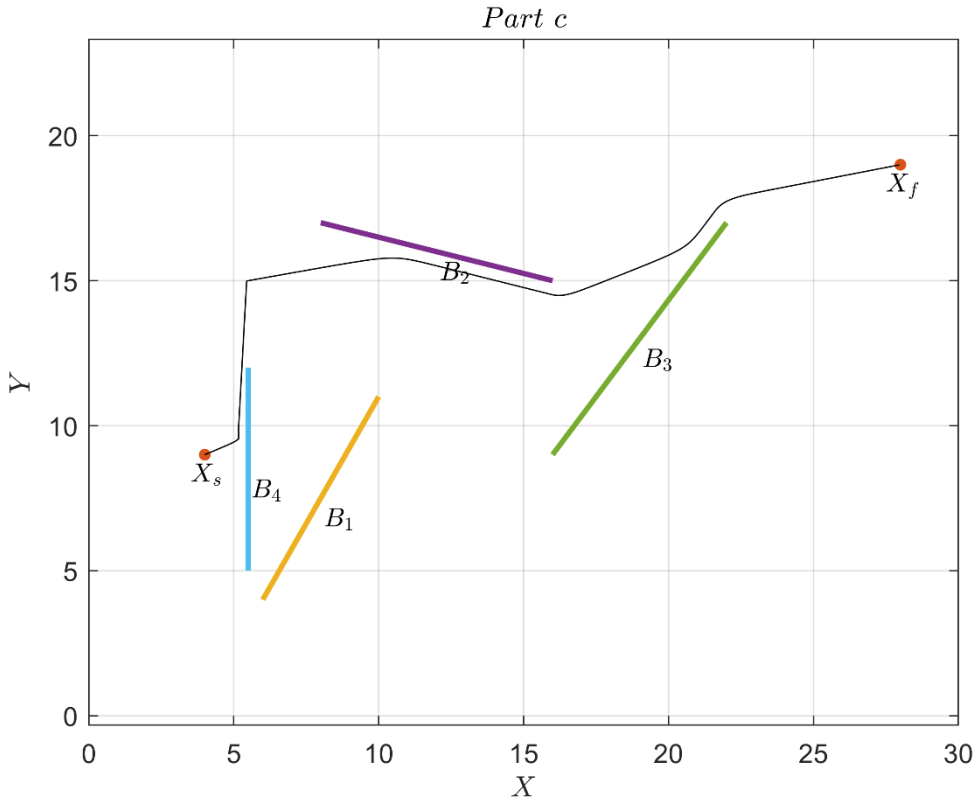
نمودار ۷) یک مسیر ممکن با قرارگیری مانع چهارم در وضعیت ۲ – پرسش ج

۳) اگر مختصات دو رأس مانع چهارم به صورت $\binom{26}{18}$ و $\binom{26}{24}$ باشد:



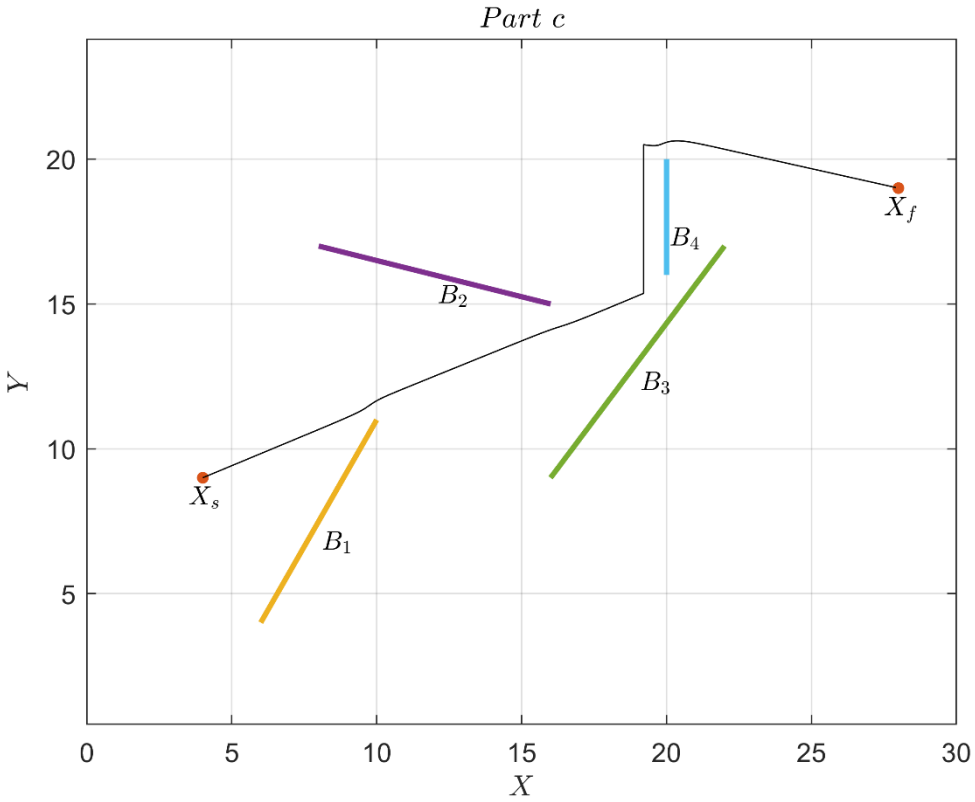
نمودار ۸) یک مسیر ممکن با قرارگیری مانع چهارم در وضعیت ۳ – پرسش ج

۴) اگر مختصات دو رأس مانع چهارم به صورت $\binom{5.5}{5}$ و $\binom{5.5}{12}$ باشد:



نمودار ۹) یک مسیر ممکن با قرارگیری مانع چهارم در وضعیت ۴ – پرسش ج

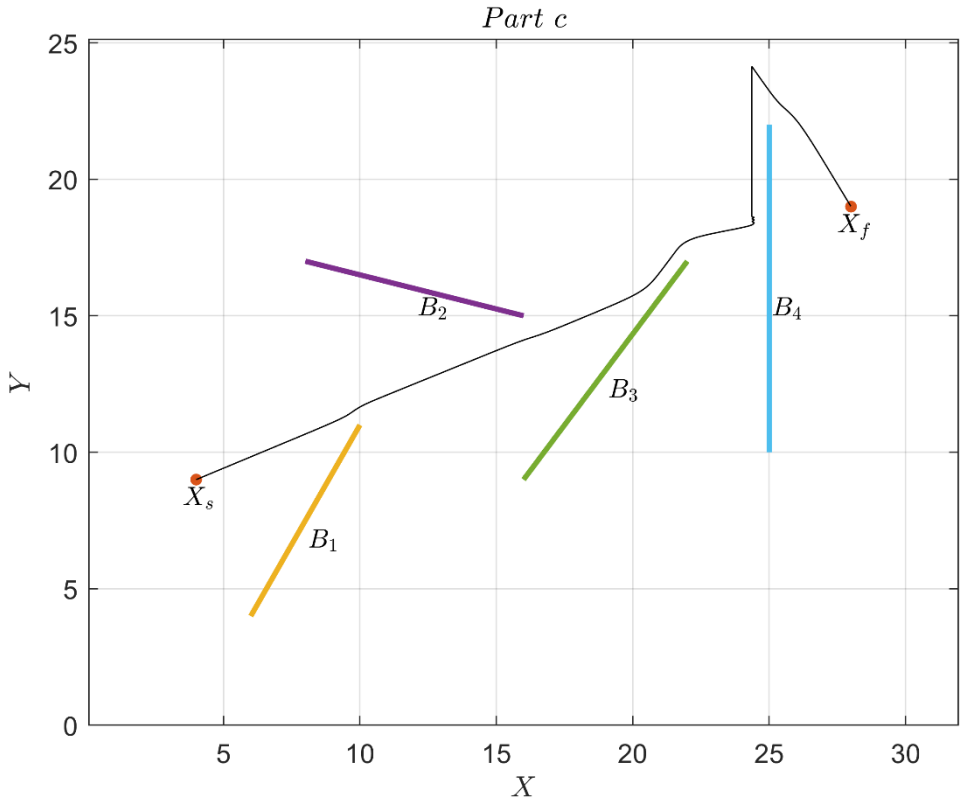
۵) اگر مختصات دو رأس مانع چهارم به صورت $\binom{20}{16}$ و $\binom{20}{20}$ باشد:



نمودار ۱۰) یک مسیر ممکن با قرارگیری مانع چهارم در وضعیت ۵ – پرسش ج

این حالت را وضعیت خاص ۲ نامیده‌ایم. در این حالت برای خروج از ناحیه‌ی کمینه‌ی محلی نمی‌توان طبق روش توضیح داده شده در بخش الگوریتم $random\ walk$ عمل نمود. در این حالت، مطابق برنامه‌ی تابع $Random\_walk.m$، یک پرش به طول مشخص (به کمک متغیر $Threshold\_RW$ که از تابع تولید مسیر به تابع $Random\_walk.m$ ارسال می شود) به موازات مانع (فقط حالت عمودی برای این تمرین لحاظ شد) و در جهت مناسب انجام می‌دهیم.

۶) اگر مختصات دو رأس مانع چهارم به صورت $\binom{25}{10}$ و $\binom{25}{22}$ باشد:



نمودار ۱۱) یک مسیر ممکن با قرارگیری مانع چهارم در وضعیت ۶ – پرسش ج

این حالت را وضعیت خاص ۳ نامیده‌ایم. در این حالت برای خروج از ناحیه‌ی کمینه‌ی محلی نمی‌توان طبق روش توضیح داده شده در بخش الگوریتم $random\ walk$ عمل نمود. در این حالت، مطابق برنامه‌ی تابع $Random\_walk.m$، یک پرش به طول مشخص (به کمک متغیر $Threshold\_RW$ که از تابع تولید مسیر به تابع $Random\_walk.m$ ارسال می شود) به موازات مانع (فقط حالت عمودی برای این تمرین لحاظ شد) و در جهت مناسب انجام می‌دهیم.

نکات کلی) ثوابت تعریف شده‌ی $count1$، $count2$، $cst3$ و $cst4$ در تابع $Random\_walk.m$ برای جلوگیری از به دام افتادن برنامه در حلقه‌ی بی‌نهایت در وضعیت‌های پیش‌بینی نشده (وضعیت خاص ۴) هستند.