

دانشکده مهندسی برق

یادگیری تقویتی در کنترل

تمرین جامع

استاد: دکتر سعید شمقدری

دانشجو: سیده ستاره خسروی

زمستان 1403

چکیده

در این تمرین به مباحث یادگیری تقویتی با رویکرد کامپیوتری و پیاده سازی در محیط Gymnasium پرداخته خواهد شد، بدین منظور 3 الگوریتم در محیط پاندول معکوس آن توسعه داده می‌شود و فرایند تست و آموزش نیز در قالب یک اپلیکیشن توسعه داده می‌شود.

در بخش دوم به کاربرد یادگیری تقویتی در کنترل بهینه پرداخته می‌شود، و توسعه فیدبک حالت بهینه برای پایدارسازی و تنظیم یک سیستم ناپایدار پیوسته پیگیری می‌شود. همچنین برای این سیستم ردیاب خطی کوادراتیک توسعه داده می‌شود.

واژه‌های کلیدی: یادگیری تقویتی، Gymnasium، LQR، LQT، Sarsa، Q Learning، Monte Carlo

# فهرست مطالب

عنوان صفحه

[فهرست مطالب ‌ب](#_Toc188625766)

[فهرست تصاویر و نمودارها ‌ج](#_Toc188625767)

[فصل 1:یادگیری تقویتی در Gymnasium 1](#_Toc188625768)

[1‌.1‌ مقدمه 2](#_Toc188625769)

[1‌.2‌ روش مونت کارلو 3](#_Toc188625770)

[1‌.3‌ روش Q Learning 11](#_Toc188625771)

[1‌.4‌ روش SARSA 15](#_Toc188625772)

[1‌.5‌ مقایسه روش‌ها 18](#_Toc188625773)

[1‌.6‌ اجرای برنامه 19](#_Toc188625774)

[فصل 2:یادگیری تقویتی در کنترل بهینه 20](#_Toc188625775)

[2‌.1‌ مقدمه 21](#_Toc188625776)

[2‌.2‌ رگولاتور خطی بهینه 21](#_Toc188625777)

# فهرست تصاویر و نمودارها

عنوان صفحه

[شکل 1: میانگین پاداش مونت کارلو در طی اپیزودهای مختلف آموزش 10](#_Toc188625756)

[شکل 2: میانگین پاداش‌های مونت کارلو در طی اپیزودهای تست 11](#_Toc188625757)

[شکل 3: میانگین پاداش روش q learning در طول آموزش 14](#_Toc188625758)

[شکل 4: نمودار پاداش در 2 اپیزود q learning فرایند تست 15](#_Toc188625759)

[شکل 5: نمودار میانگین پاداش sarsa در طول اپیزودهای آموزش 17](#_Toc188625760)

[شکل 6: پاداش دریافتی sarsa در طول 5 اپیزود تست 18](#_Toc188625761)

[شکل 7: نمودار همگرایی مقادیر k در on policy IRL 27](#_Toc188625762)

[شکل 8: نمودار همگرایی k پس از کاهش بازه نویز در on policy IRL 27](#_Toc188625763)

[شکل 9: متغیرهای حالت پس از اعمال سیاست کنترلی بهینه 28](#_Toc188625764)

[شکل 10: نمودار حالات سیستم 31](#_Toc188625765)

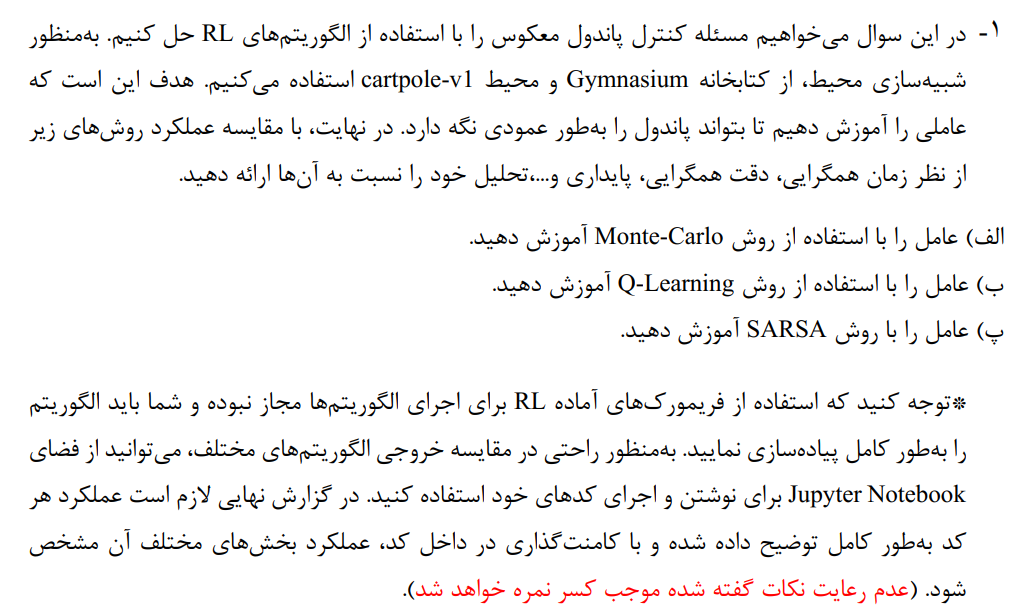
# یادگیری تقویتی در Gymnasium

## مقدمه

در این بخش به سوال اول تمرین جامع به طور کامل پاسخ داده می‌شود، عملکرد مدل‌ها تحلیل می‌گردد، همچنین عملکرد مدل‌ها در فایل‌های ویدیویی پیوست برای مشاهده قرار داده شده است.

**توجه!**

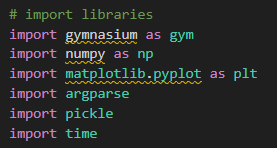
برای مشاهده بدون مشکل ویدیوها، از Player هایی مانند KMPlayer، PotPlayer و ... استفاده شود. از Windows Media Player استفاده نکنید.

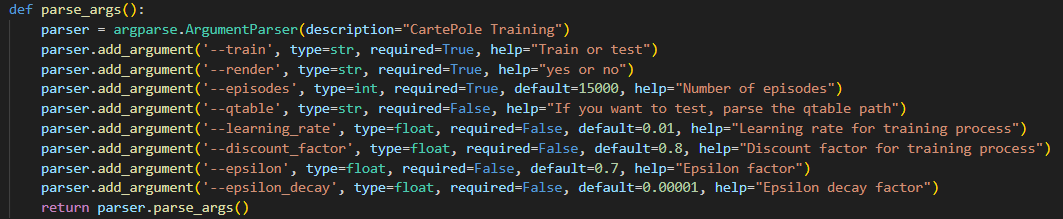
**صورت سوال:**

## روش مونت کارلو

در این بخش به قسمت الف سوال اول پاسخ داده می‌شود.

به منظور پیاده سازی این بخش، کدی به نام CartePoleOnMonteCarlo.py توسعه داده شد. در این کد روش On Policy Monte Carlo پیاده سازی شده است. قدم به قدم اجزای کد را توضیح می‌دهیم.

ابتدا مطابق تصویر زیر کتابخانه‌های مورد نیاز برای پیاده سازی الگوریتم را فراخوانی می‌کنیم.

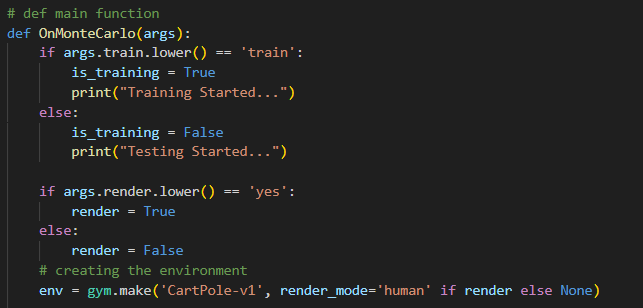
سپس با استفاده از کتابخانه argparse، تابعی نوشته شد، که کاربر بتواند کد را با پاس دادن پارامترهای مهم آن در ترمینال سیستم خود اجرا کند.

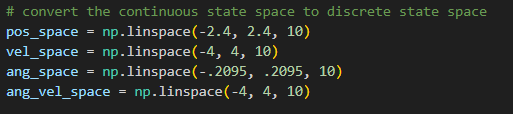
جزئیات پارامترها به شرح زیر است:

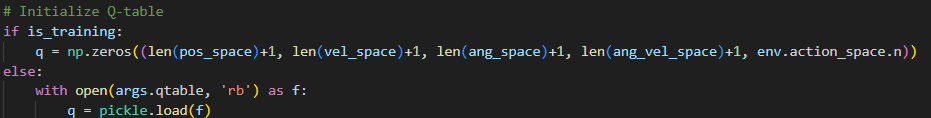
1. Train: انتخاب می‌کنید که فرآیند تست انجام شود یا آموزش.
2. Render: با استفاده از این پارامتر مشخص می‌کنید که در هنگام آموزش و یا تست رندر انجام شود یا خیر.
3. Episodes: با استفاده از این پارامتر تعداد اپیزودهای لازم توسط کاربر مشخص می‌شود.
4. Qtable: با استفاده از این پارامتر، مسیر Q Table ذخیره شده برای الگوریتم Monte Carlo به کد پاس داده می‌شود که در هنگام تست از آن استفاده می‌گردد.
5. Learning\_rate: با استفاده از این پارامتر نرخ یادگیری مشخص می‌گردد، که مقدار پیش‌فرض آن 0.01 لحاظ شده است.
6. با استفاده از پارامتر discount\_factor نرخ تخفیف تنظیم می‌شود. مقدار پیش‌فرض آن برابر با 0.8 است.
7. با دو پارامتر نهایی نیز مقدار epsilon و نرخ کاهش آن تنظیم می‌شود. که مقادیر پیش‌فرش آن‌ها به ترتیب 1 و 0.00001 است.

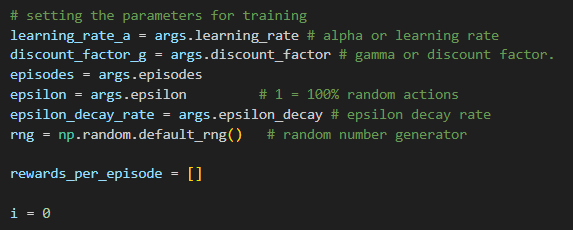
سپس تابع OnMonteCarlo برای کدنویسی فرایند آموزش و تست توسعه داده شد. مطابق کد زیر، ابتدا با استفاده از آرگومان‌هایی که کاربر به سیستم می‌دهد، فرایند اصلی که آموزش باشد یا تست مشخص می‌شود، سپس مشخص می‌گردد که رندر صورت پذیرد یا خیر.

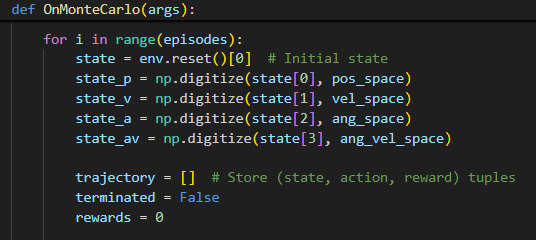
در نهایت محیط پاندول معکوس ساخته می‌شود.

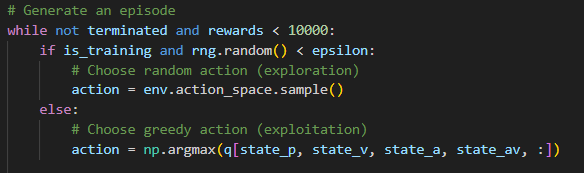


محیط CartePole محیطی است با فضای حالت (state یا Observation) پیوسته با که دارای 4 پارامتر است. این فضا شامل اطلاعاتی درخصوص پوزیشن و سرعت ارابه، زاویه میله و سرعت زاویه‌ای آن است. با توجه به اینکه این فضا پیوسته است لازم است برای پیاده سازی الگوریتم‌ها گسسته شود. با استفاده از کد زیر فضای حالت سیستم را گسسته می‌کنیم.

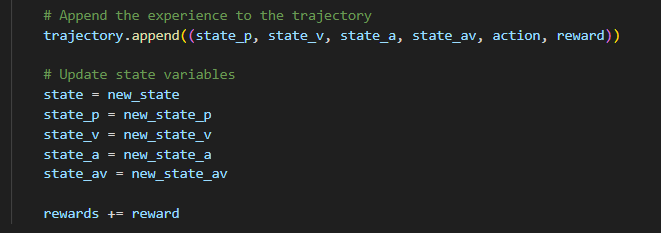
در ادامه براساس اینکه در فرایند آموزش هستیم Qtable ابتدایی را تشکیل می‌دهیم. ابعاد آن متناسب با فضای حالت است، که برابر است با 11\*11\*11\*2. این مقدار 2 درواقع نشانگر فضای عمل است که گسسته است. اگر درفرایند تست باشیم، Qtable ذخیره شده را برای اجرا فرامی‌خوانیم.

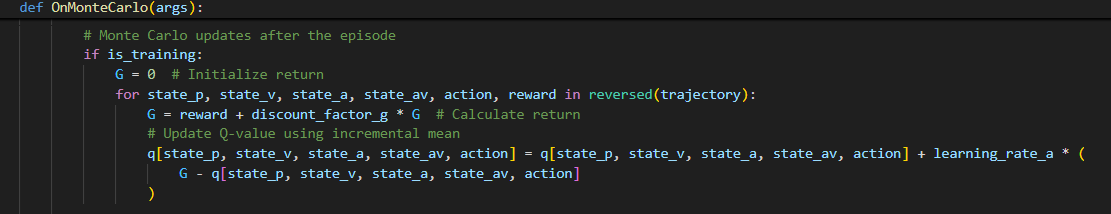
با استفاده از کد زیر تنظیماتی که کاربر وارد کرده، با استفاده از آرگومان‌های مربوطه، در متغیرهای محلی متناظر ذخیره می‌کنیم.

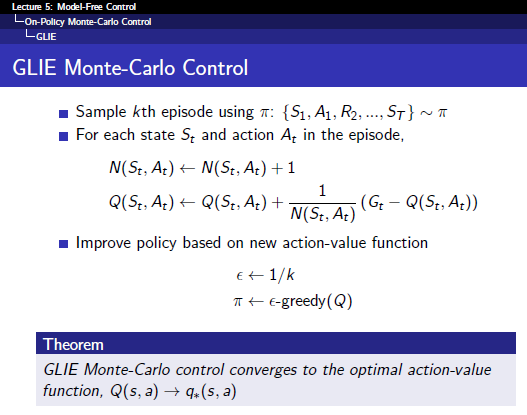
در حلقه آموزش، ابتدای هر اپیزود محیط reset می‌شود، حالت نیز در متغیرهای محلی ذخیره می‌شود، تا برای بروزرسانی تابع ارزش استفاده شوند. یک متغیر دیگر به نام trajectory تعریف می‌شود، زیرا لازم است یک مسیر کامل تا انتها تولید شود.

همانطور که گفتیم برای مونت کارلو لازم است یک اپیزود تا انتها تولید شود، به همین منظور از حلقه درونی استفاده می‌کنیم. در این حلقه ابتدا یک عدد تصادفی تولید می‌گردد، اگر این عدد از مقدار epsilon کمتر باشد، عملی تصادفی انتخاب می‌شود. در غیر این صورت ویا اگر در فرایند تست باشیم، به صورت greedy عمل خواهیم کرد.

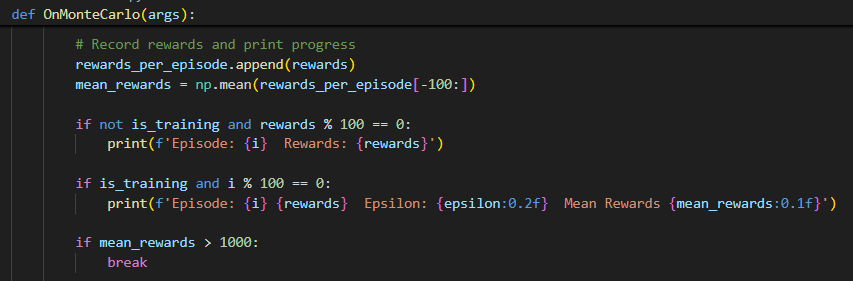
پس از اعمال این عمل به محیط، حالت جدید دریافت می‌شود، گسسته سازی شده و در متغیر محلی مربوطه ذخیره می‌شود.

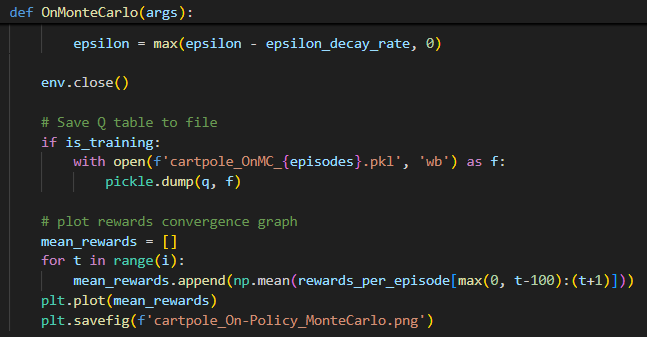
سپس این حالت جدید، به همراه عمل انجام شده و پاداش دریافتی به متغیر trajectory اضافه می‌شود.

پس از تولید trajectory تا terminal، از حلقه درونی خارج می‌شویم. اگر در فرایند آموزش باشیم، Qtable به صورت زیر به روز می‌شود. ابتدا برای هر تک المان درون trajectory، که شامل 4 حالت، یک عمل و reward مربوطه است، میزان return محاسبه می‌شود. سپس Qtable به صورت بازگشتی به روز می‌گردد.

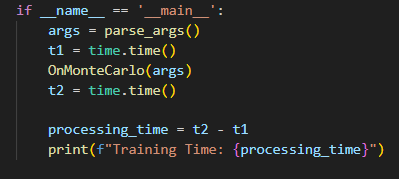
نحوه به روز رسانی تابع ارزش حالت و عمل مطابق معادله زیر است:

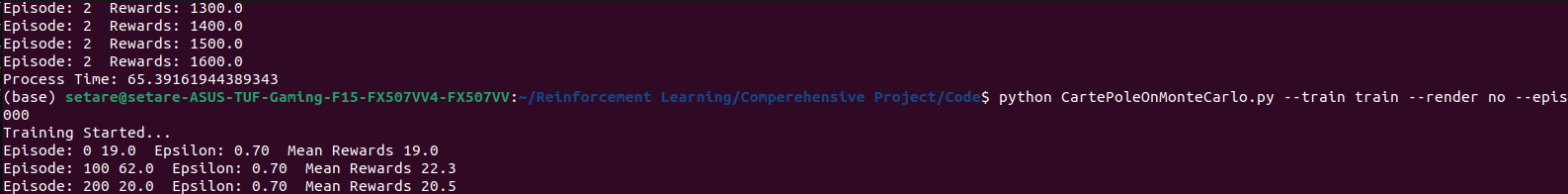
این معادله مربوط به GLIE Monte Carlo Control[[1]](#footnote-1) است که روشی On Policy است.

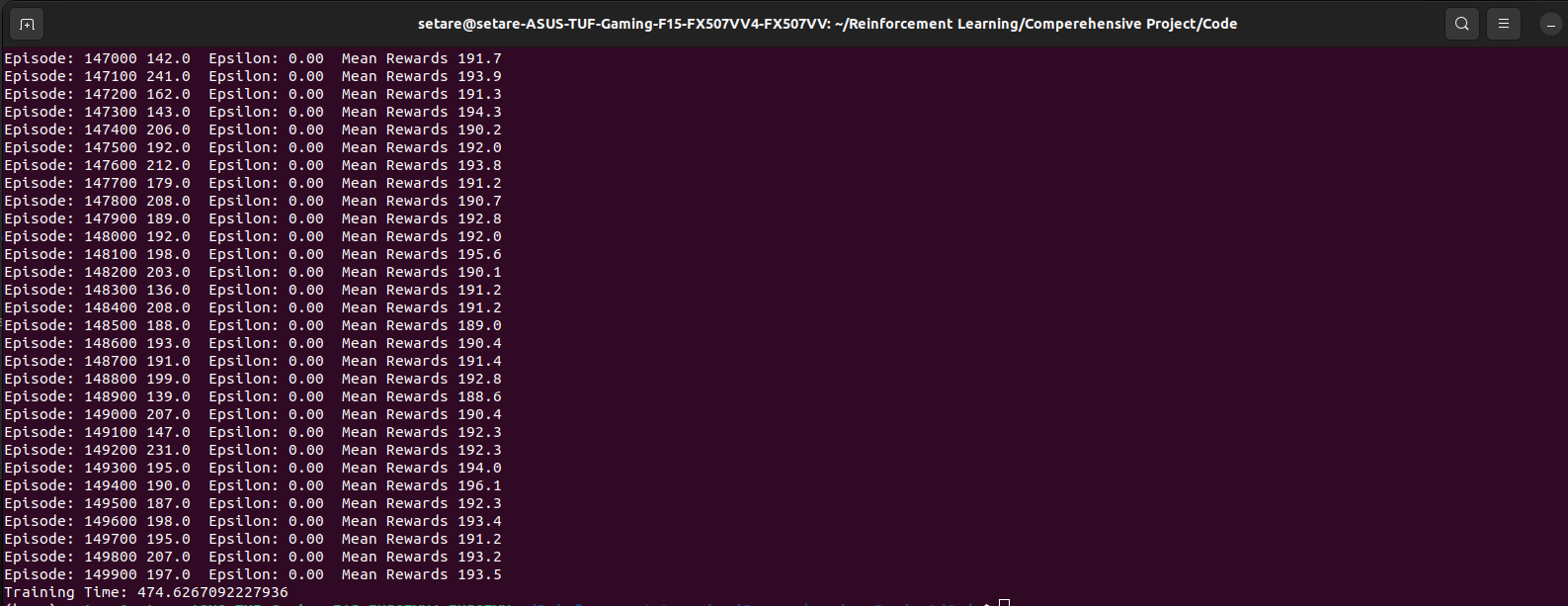
سپس میانگین پاداش‌ها بدست می‌آید، و مقادیر در ترمینال چاپ می‌شود. اگر میزان پاداش در یک اپیزود بیش از 1000 شود، این اپیزود قطع می‌شود.

سپس اپسیلون در انتهای اپیزود کاهش می‌یابد. پس از اتمام آموزش Qtable با استفاده از کتابخانه pickle ذخیره می‌گردد. نمودار تغییرات میانگین پاداش نیز رسم می‌گردد.

در نهایت تابع با پاس دادن آرگومان‌ها و تنظمیات اجرا، اجرا می‌گردد، زمان اجرای فرایند آموزش و یا تست نیز اندازه گیری شده جهت مقایسه در ترمینال چاپ می‌شود.

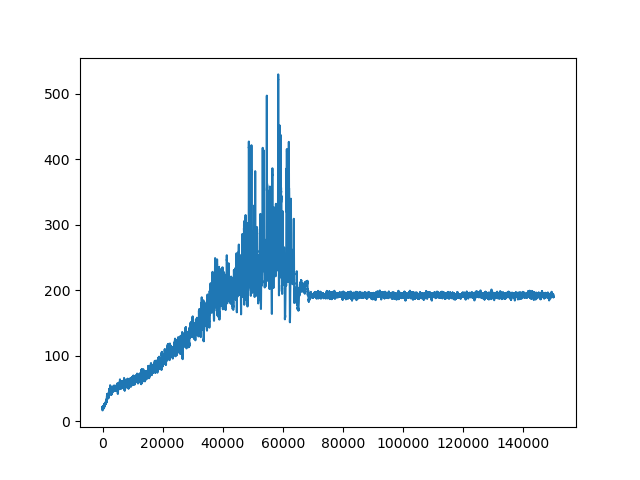


توجه! برای آموزش هر سه روش از پارامترهای پیش‌فرضی که در توضیحات ابتدایی ذکر شد استفاده شده است و تعداد اپیزود‌های تمام آموزش‌ها 150000 است.

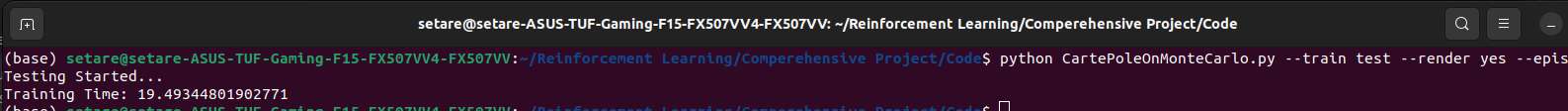
پس از اتمام اجرا، زمان اجرا نیز به صورت زیر در ترمینال چاپ شد.

همانطور که مشاهده می‌شود، زمان اجرای فرایند آموزش 474 ثانیه است.

نمودار میانگین پاداش در شکل 1 رسم شده است. به لحاظ همگرایی ظاهرا عملکرد حین آموزش خوب بوده است. اما لازم است Q Table نهایی آن با اجرا ارزیابی شود.

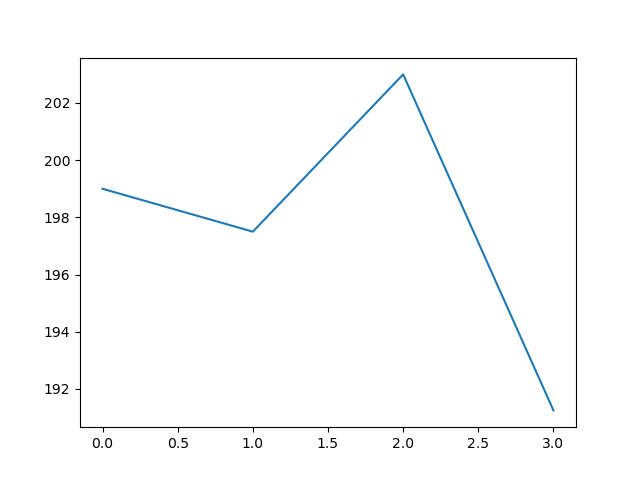


شکل 1: میانگین پاداش مونت کارلو در طی اپیزودهای مختلف آموزش

با استفاده از کد زیر نیز، فرایند تست با فعالسازی رندر اجرا می‌گردد.

زمان اجرا 20 ثانیه است، و نمودار میانگین پاداش در طی اپیزودهای تست به صورت شکل 2 است (5 اپیزود)

باتوجه به زمان اجرای تست و میانگین پاداش‌های آن، عملکرد مدل ضعیف است. ویدیوی عملکرد آن نیز در پیوست موجود است.



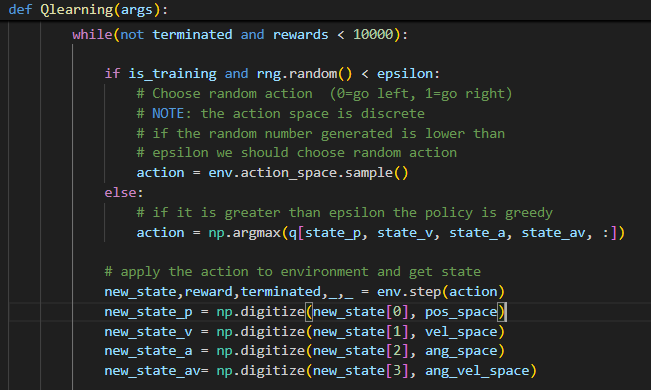
شکل 2: میانگین پاداش‌های مونت کارلو در طی اپیزودهای تست

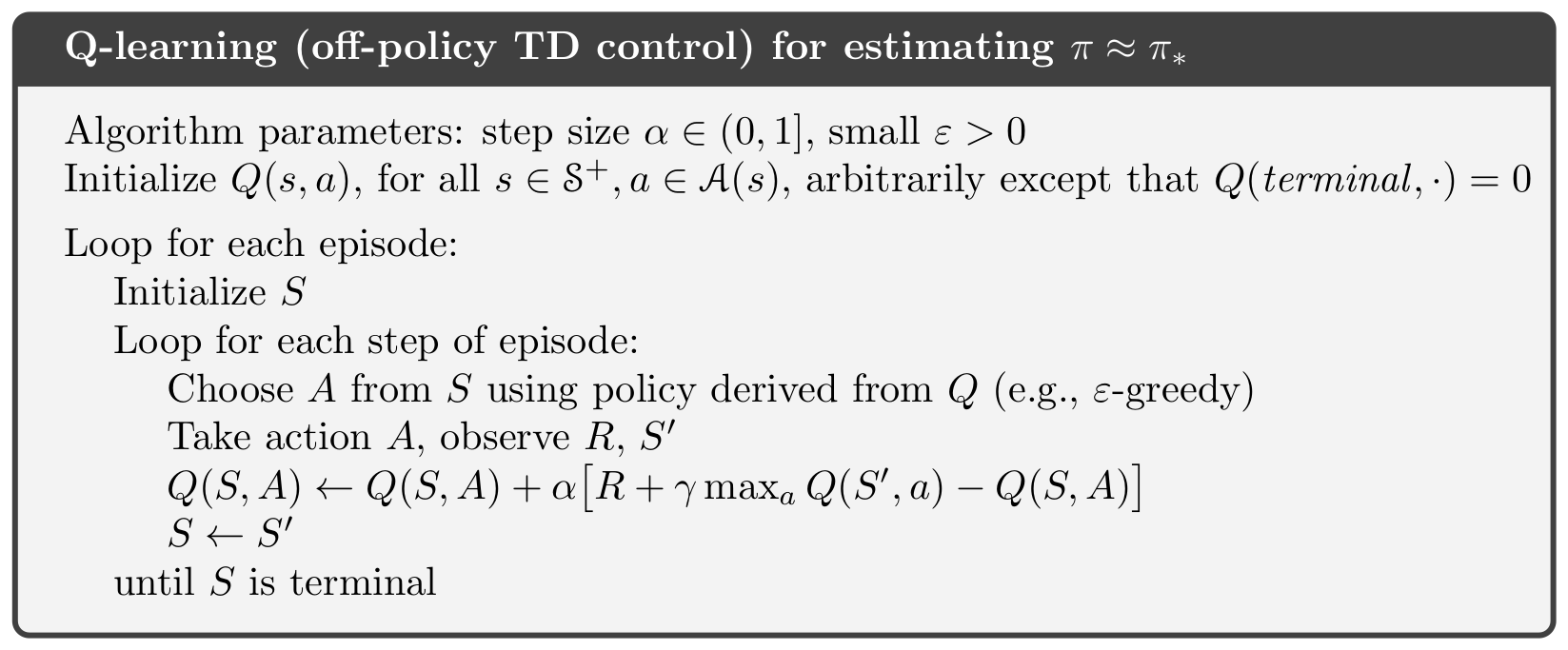
در قسمت جمع‌بندی به مقایسه روش‌های مختلف نیز می‌پردازیم.

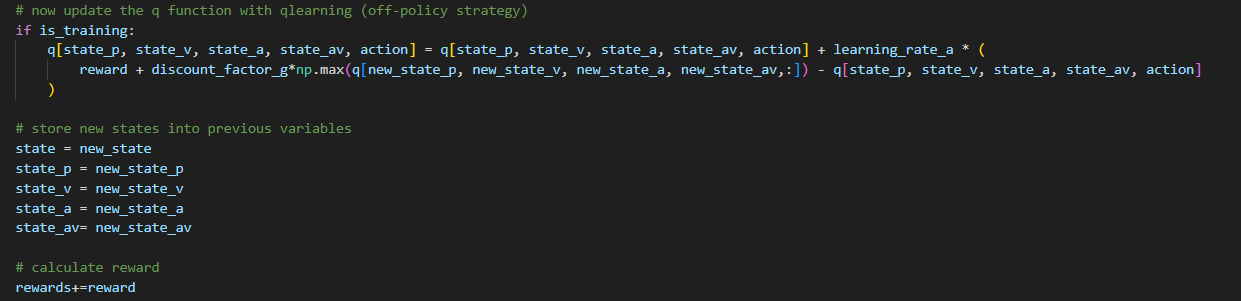
## روش Q Learning

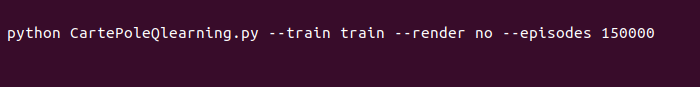
در این قسمت به منظور پیاده سازی روش Q Learning از کدی مشابه با کد مورد استفاده در قسمت قبل استفاده شد. پارامترها مشابه قبل است، کد این بخش تا خطوط حدود 80 مشابه کد مونت کارلو است. در این بخش فقط متغیر trajectory تعریف نمی‌گردد زیرا نیازی به ذخریه سازی کامل یک اپیزود به آن شیوه نیست.

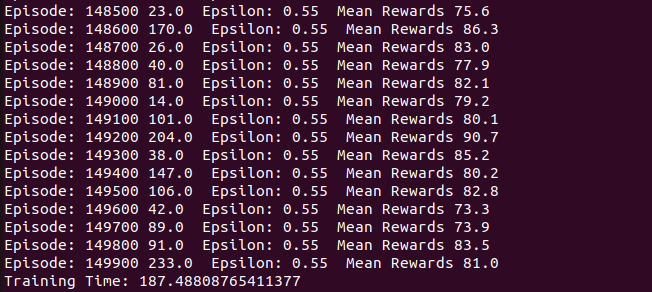
در حلقه درونی، ابتدا بر اساس عدد تصادفی تولید شده، تصمیم گیری ‌می‌شود که عمل به صورت تصادفی و یا به صورت حریصانه انتخاب شود. سپس این عمل به محیط اعمال می‌شود، حالت جدید دریافت، گسسته ودر متغیر محلی متناظر مانند مونت کارلو ذخیره می‌شود.



تفاوت اصلی این کد با کد قبل در نحوه بروز رسانی تابع ارزش حالت عمل است. با الگو گیری از رابطه زیر:

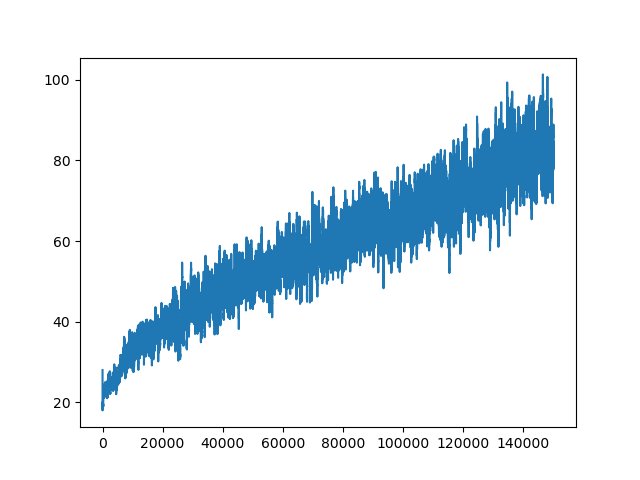
بروز رسانی را به صورت زیر پیاده سازی کردیم، سپس حالات جدید جایگزین حالات قبلی می‌شوند.

سایر قسمت‌های کد نیز مشابه کد مونت کارلو است. برای آموزش از دستور زیر در ترمینال استفاده می‌کنیم.

در آموزش با Q Learning نرخ کاهش epsilon را نیز کاهش دادیم تا epsilon به صفر نرسد.

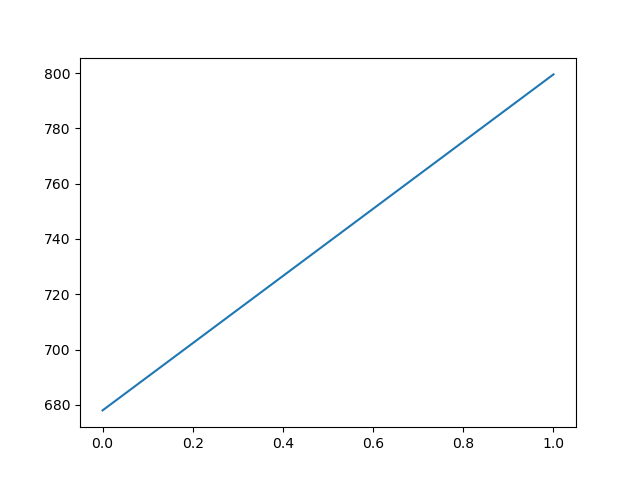
همانطور که مشاهده می‌شود، فرایند آموزش بیش از 2 برابر سریعتر از مونت کارلو اجرا می‌گردد.

در شکل 3، نمودار میانگین پاداش در طول اپیزودهای آموزش رسم شده است. همانطور که مشاهده می‌شود روندی صعودی دارد، دلیل اینکه به مقدار بیشینه همگرا نشده است و همچنان در حال افزایش است این است که مقدار epsilon در انتها به حددود 0.5 می‌رسد و همچنان از جست و جو نیز در کنار انتخاب حریصانه استفاده می‌گردد. در مونت کارلو چون نرخ کاهش epsilon کمتر لحاظ شده بوده است، به مقدار بیشینه‌ای که می‌توانسته برسد، همگرا می‌شود.

مانند بخش قبل، عملکرد این روش ارزیابی گردید. خروجی تست آن به صورت زیر است:

شکل 3: میانگین پاداش روش q learning در طول آموزش

همانطور که مشاهده می‌شود، زمان اجرای فقط 2 اپیزود آن 65 ثانیه است و پاداش در انتها به بیش از 1000 رسیده است. نمودار مربوطه در شکل 4 موجود است.

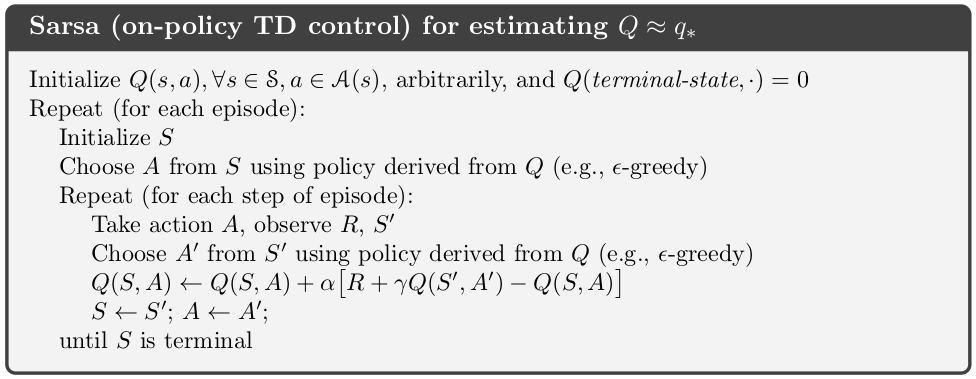
ویدیوی عملکرد آن نیز در پیوست موجود است، در مجموع عمکلرد این مدل مناسب است.

شکل 4: نمودار پاداش در 2 اپیزود q learning فرایند تست

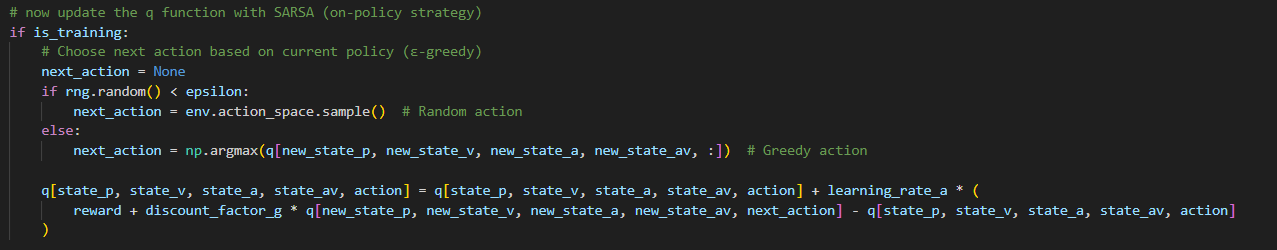
## روش SARSA

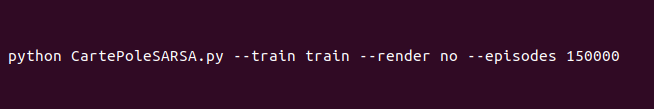
در این بخش نیز کد مربوط به SARSA پیاده سازی گردید.

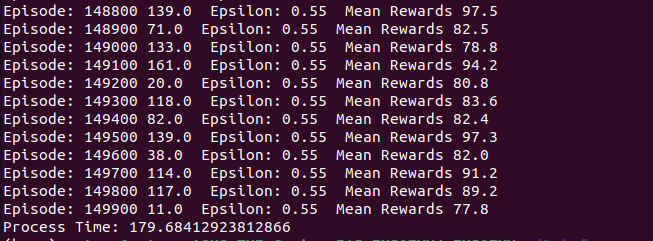
**شرایط آزمایش (پارامترها و تعداد اپیزود) در این بخش کاملا مشابه قسمت قبلی، Q Learning است.**

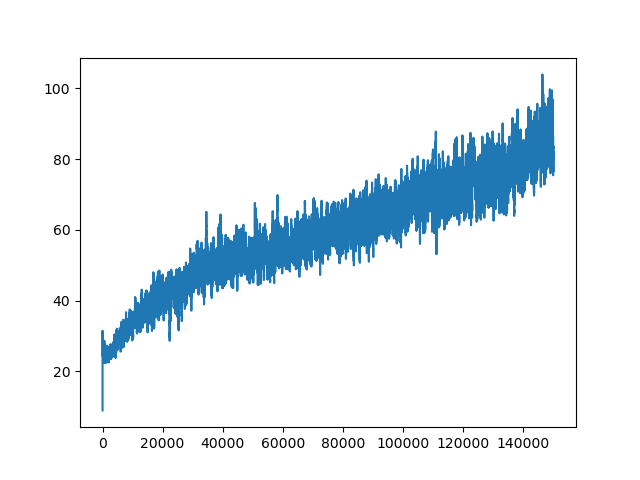
کد این بخش نیز با استفاده از کد Q Learning نوشته شد، تنها تفاوت پیاده سازی SARSA در نحوه‌ی پیاده سازی بروزرسانی تابع ارزش حالت و عمل است.

با استفاده از رابطه بالا، بروز رسانی را تغییر دادیم.

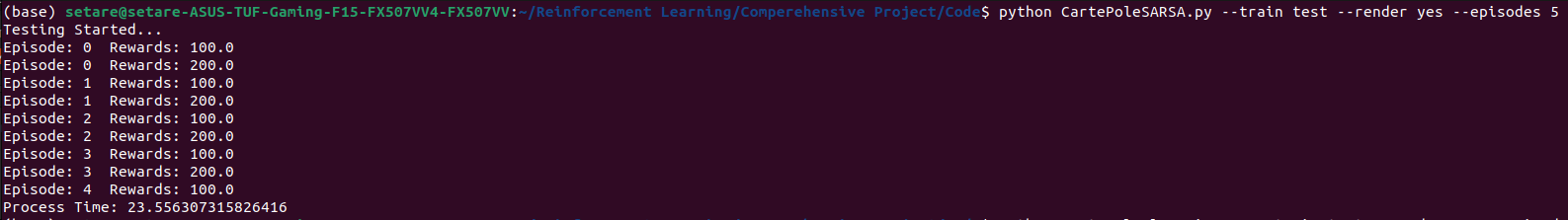
مطابق الگوریتم، در ابتدا، عمل انتخاب می‌شود، پس از اعمال آن به محیط، عمل بعدی نیز براساس سیاست e-greedy صرفا انتخاب می‌شود ولی اعمال نمی‌شود، صرفا از آن برای بروز رسانی تابع ارزش حالت و عمل استفاده می‌شود.

سایر قسمت‌های کد مشابه قبل است. از دستور زیر برای اجرای فرایند آموزش در ترمینال استفاده کردیم.

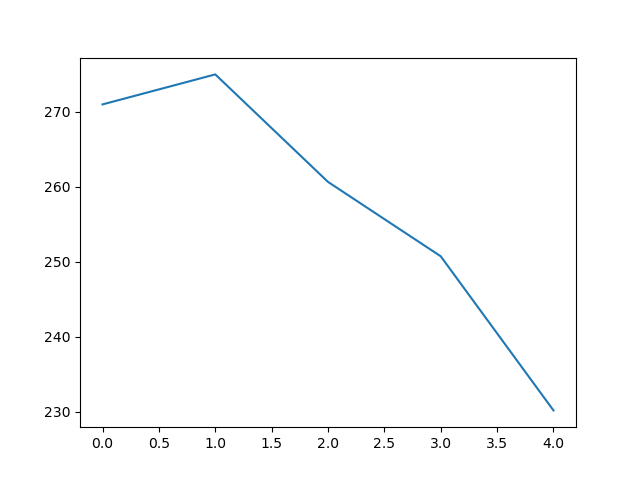
فرایند آموزش آن حدود 9 ثانیه سریعتر از Q learning اجرا گردید.

نمودار میانگین پاداش آن در طول اپیزودهای آموزش، روندی مشابه Q learning دارد زیرا نرخ کاهش epsilon را به گونه‌ای تنظیم کردیم که در انتها به حدود 0.5 و نه به صفر نرسد.

شکل 5: نمودار میانگین پاداش sarsa در طول اپیزودهای آموزش

تست آن نیز به صورت زیر انجام شد:

از روی مدت زمان اجرای 5 اپیزود و مقادیر پاداش‌های دریافتی مشخص است که عملکرد این مدل وضعیت مناسبی ندارد. (ویدیوی پیوست مشاهده شود)

نمودار پاداش دریافتی آن در طول تست به شکل زیر است:

شکل 6: پاداش دریافتی sarsa در طول 5 اپیزود تست

شکل6 نیز مطابق توضیحات قبلی است که در خصوص عملکرد این مدل بیان گردید.

## مقایسه روش‌ها

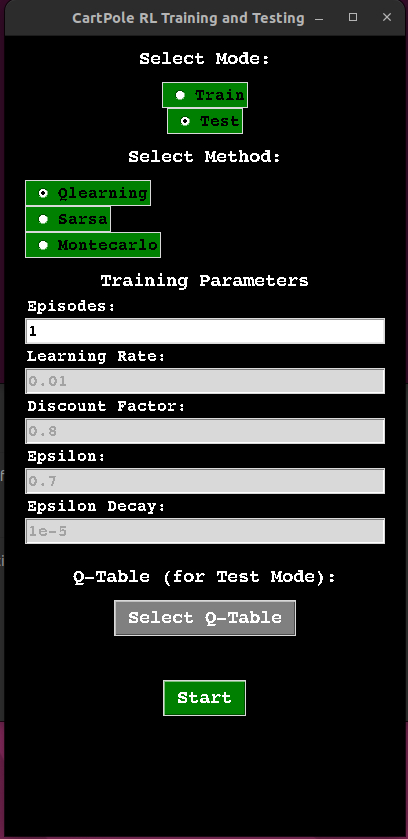
در این بخش روش‌ها را به لحاظ پارامترهای مختلف مقایسه می‌کنیم.

1. زمان همگرایی: با وجود تفاوت در نرخ کاهش epsilon، سرعت همگرایی مونت کارلو بسیار کندتر از دو روش دیگر است.
2. دقت همگرایی: باتوجه به عملکرد مدل‌ها و نمودارهای رسم شده، دقت همگرایی Qlearning از دو روش دیگر بیشتر است، مطابق تئوری درس انتظار می‌رود Qlearning به راه حل بهینه، بهتر از دو روش دیگر، برسد زیرا ماهیتی Off Policy دارد.
3. پایداری: به لحاظ پایداری، Q learning به دلیل رسیدن به جوابی بهینه‌تر نسبت به سایر روش‌ها باعث پایدارسازی بهتر سیستم می‌گردد.

## اجرای برنامه

برای اجرای ساده تر کدها و مشاهده عملکرد آن‌ها، دو کد تحت نام‌های main.py و app.py توسعه داده شده است. کد اول لازم است با وارد کردن آرگومان‌ها در ترمینال اجرا شود (مانند سه کد قبلی) اما با اجرای دستور ساده‌ه زیر در ترمینال:

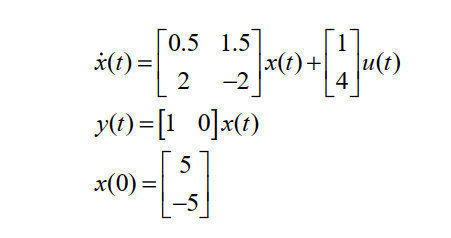
python main.py

پنجره GUI ساده‌ای که توسعه داده شده است، باز میگردد.

در این پنجره، می‌توانید انتخاب کنید که چه فرایندی انجام شود، اگر آموزش را انتخاب کنید، لازم است پارامترهای موثر در آموزش را نیز تنظیم کنید. اگر تست را انتخاب کنید صرفا لازم است تعداد اپیزود‌ها را تنظیم کنید و مسیر Qtable را نیز انتخاب کنید. با انتخاب هر یک از 3 روش، می‌توانید عملکرد روش‌ها را مشاهده کنید.

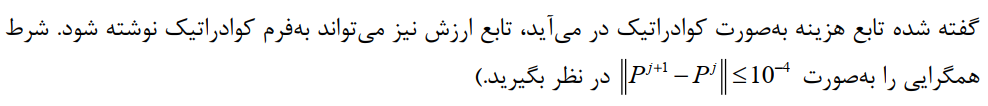
# یادگیری تقویتی در کنترل بهینه

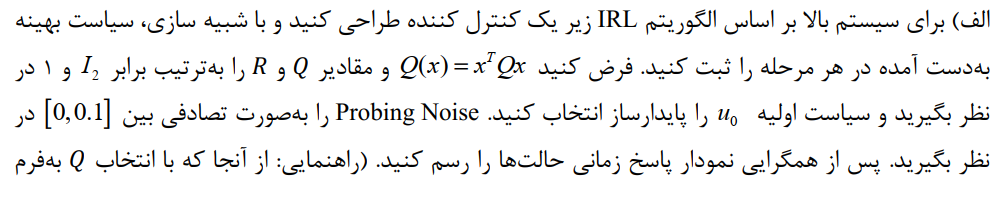
## مقدمه

در این بخش با یک سیستم خطی پیوسته اما ناپایدار روبرو هستیم. لازم است برای آن رگولاتور درجه دوم خطی و ردیاب درجه دوم خطی بهینه طراحی شود.

## رگولاتور خطی بهینه

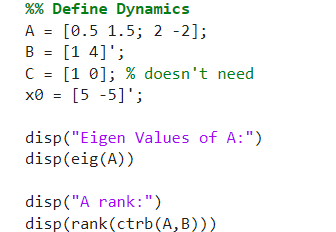
در بخش 2.2 به سه قسمت ابتدایی سوال دوم که مربوط به طراحی کنترل کننده برای مسئله LQR است پاسخ می‌دهیم.

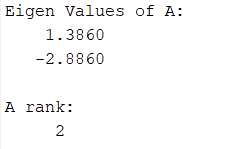
**صورت سوال بخش الف:**



**پاسخ بخش الف:**

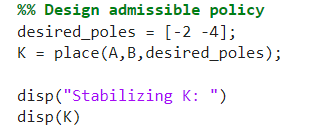
ابتدا برای حل سوال دینامیک سیستم را بررسی می‌کنیم.

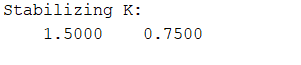
با استفاده از کد زیر، ابتدا دینامیک سیستم را تعریف می‌کنیم، سپس مقادیر ویژه ماتریس A را برای تحلیل پایداری سیستم نمایش می‌دهیم، سپس به لحاظ کنترل پذیری، با استفاده از دستور ctrb سیستم را بررسی می‌کنیم، علیرغم اینکه سیستم دارای قطب ناپایدار است اما ماتریس کنترل پذیری آن رنک کامل است، بنابراین می‌توان بدون نگرانی برای این سیستم کنترلر طراحی نمود.

خروجی کد بالا به صورت زیر است:

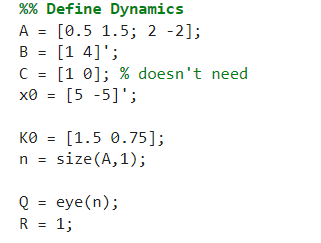
سپس برای اینکه بتوانیم با روش IRL، برای سیستم کنترل کننده طراحی کنیم، نیاز داریم یک سیاست پایدار ساز اولیه نیز بدست آوریم. در اینجا محل قطب‌های اولیه مطلوب را در 2- و 4-، انتخاب می‌کنیم، سپس با دستور place بهره فیدبک حالتی که شرایط را براورده می‌کند بدست می‌آوریم.

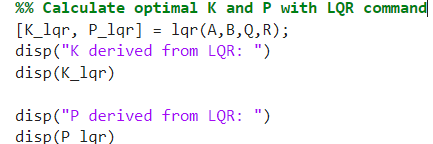
بدین منظور از کد زیر استفاده می‌کنیم.

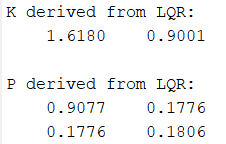


بهره فیدبک حالت به صورت زیر بدست می‌آید:

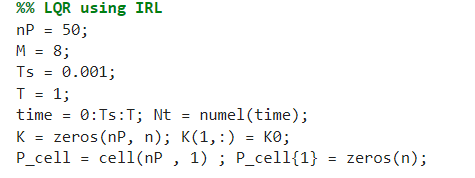
که از آن برای پیگیری فرایند طراحی کنترل کننده بهینه با رویکرد IRL بهره خواهیم برد.

با استفاده از کد زیر دینامیک سیستم و تابع هزینه را تعریف می‌کنیم.

سپس با استفاده از دستور lqr مقادیر k و p بهینه را بدست می‌آوریم.

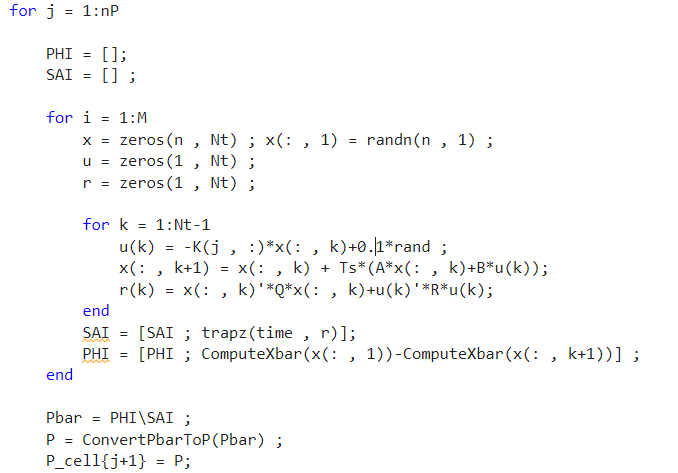
که مقادیر آن برابر است با:

حال لازم است برای انجام فرایند آموزش، تنظیمات آن را تعیین کنیم. تعداد تکرارها را برابر با 50 لحاظ می‌کنیم، باتوجه به اینکه ابعاد ماتریس P 2\*2 است، باید 3 داده برداری صورت بگیرد که جهت اطمینان آن را مساوی 8 قرار می‌دهیم. برای شبیه سازی قسمت انتگرالی نیز باید از روش‌های عددی و تقریب اویلر استفاده کنیم. بدین منظور نرخ نمونه برداری را برابر با 0.001 انتخاب می‌کنیم. سپس سایر پارامترها جهت بروز رسانی ارزش و سیاست را تنظیم می‌کنیم، اولین المان سیاست را نیز، که سیاست اولیه می‌باشد، برابر با بهره k ای قرار می‌دهیم که با دستور place بدست آوردیم.

سپس برای مقادیر مختلف ماتریس‌های P نیز، متغیری را به صورت سلولی تعریف کردیم که بتوانیم مقادیر مختلف این ماتریس را در سلول‌های آن ذخیره و برای شرط توقف استفاده کنیم.

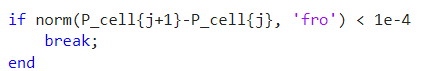
حال حلقه‌های برنامه را می‌نویسیم، حلقه بیرونی به تعداد کل تکرارها یعنی 50 بار قرار است اجرا شود، ابتدا ماتریس‌های SAI و PHI را تعریف می‌کنیم، از آن‌ها برای اجرای LS قرار است استفاده گردد.

سپس در حلقه دوم داده برداری قرار است صورت پذیرد که درون آن به وسیله حلقه سوم دینامیک سیستم و ورودی کنترلی به علاوه نویز گفته شده در صورت سوال شبیه سازی و مقادیر پاداش با تقریب اویلر محاسبه می‌شود.

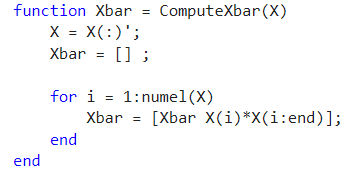
پس از اتمام حلقه سوم، درون حلقه دوم مقادیر ماتریس‌های SAI و PHI محاسبه می‌گردد. برای محاسبه انتگرال پاداش، از روش ذوزنقه و دستور trapz استفاده می‌کنیم. باید برای محاسبه و استفاده از ls بردار x به فرم xbar یا همان ضرب کرونکر خودش در خودش تبدیل شود (توسط تابعی که در انتهای کد وجود دارد این تبدلی صورت می‌گیرد)، پارامتر Pbar که فرم برداری P با پارامترهای مستقل است (یعنی 3 درایه دارد) از تقسیم PHI و SAI بدست می‌آید که از روی آن باید ماتریس P را محاسبه کنیم. توضیحات داده شده به صورت زیر پیاده سازی گردید:

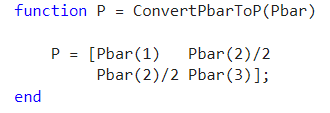
پس از آن سیاست به صورت زیر بروزرسانی می‌گردد.

شرط توقف مطابق صورت سوال به صورت زیر پیاده سازی گردید.



در ادامه مقادیر K بهینه حاصل از IRL، چاپ و روند همگرایی آن رسم می‌گردد.

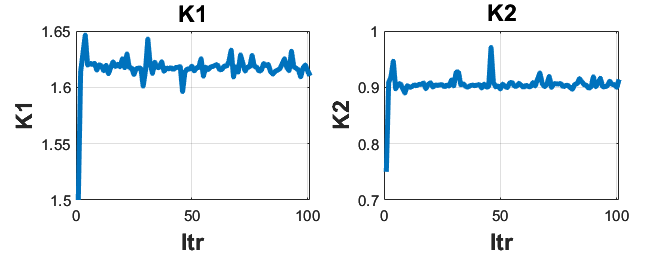
برای محاسبه حاصل ضرب کرونکر x در خودش از تابع زیر استفاده کردیم.

و برای تبدیل P به فرم ماتریسی آن، از تابع زیر استفاده گردید.

حال کد را به طور کامل اجرا می‌کنیم.

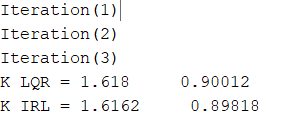
شرط توقف با اجرای کد برقرار نگردید، مقادیر سیاست بهینه در مقایسه با LQR به صورت زیر بدست آمد:

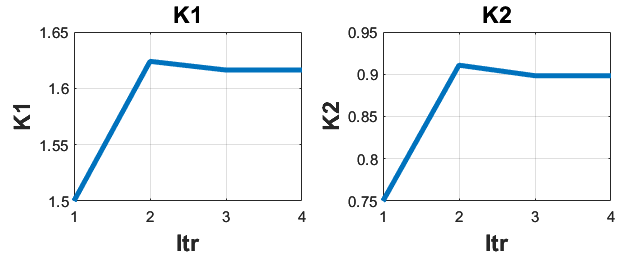
مقدار بدست آمده بسیار نزدیک حالت بهینه است. می‌توانیم تعداد تکرارها را بیشتر کنیم. پس از صد تکرار داریم:

نمودار همگرایی نیز به صورت زیر بدست می‌آید:

شکل 7: نمودار همگرایی مقادیر k در on policy IRL

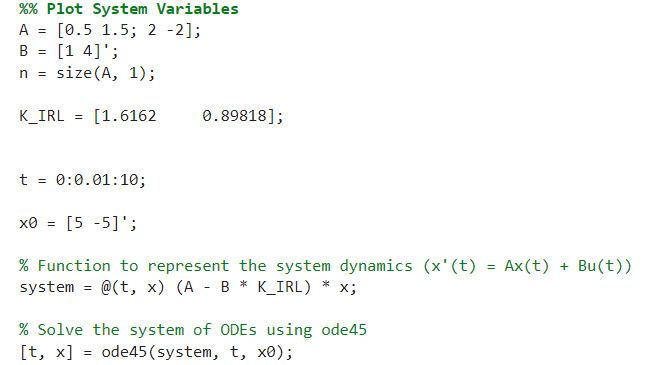
می‌توانیم نویز اضافه شده را کوچکتر کنیم، ضریب آن‌ را بجای 0.1، 0.01 قرار می‌دهیم.

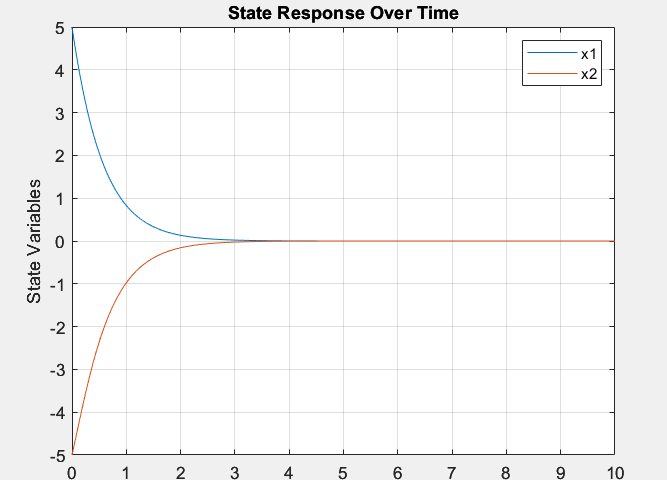
در صورت کاهش بازه‌ی تغییرات نویز، در 3 تکرار شرط توقف اغنا می‌گردد و نتایج به صورت زیر خواهد بود که بسیار به نتیجه LQR نزدیک است:

و همگرایی K در 3 تکرار به صورت زیر خواهد بود:

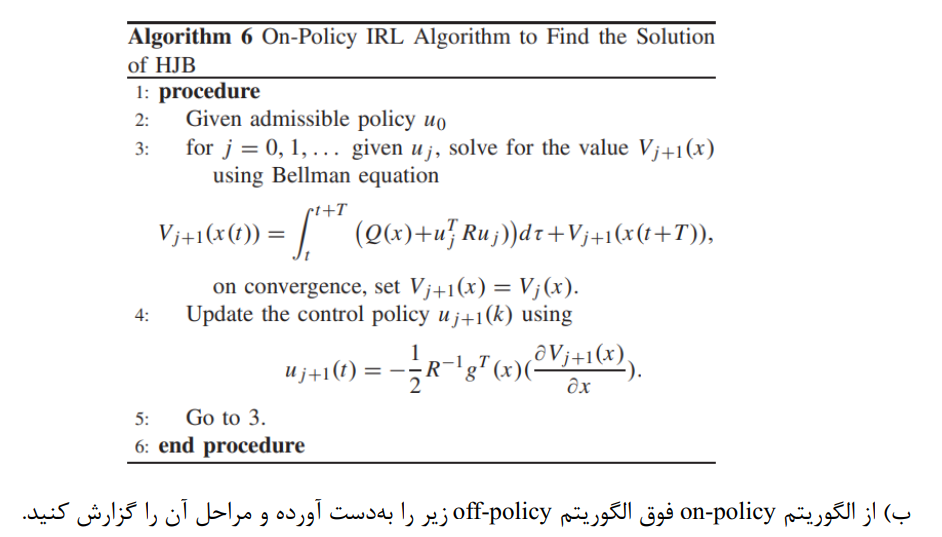
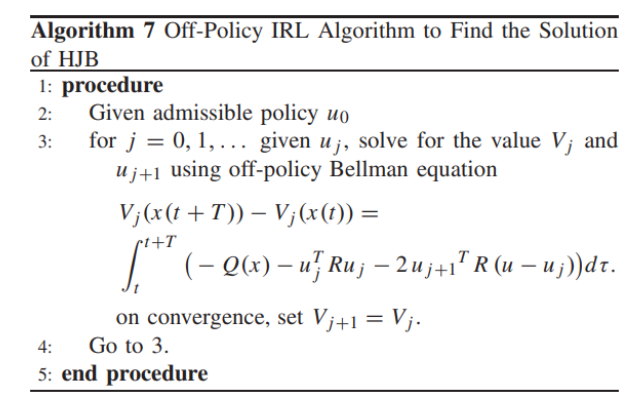
شکل 8: نمودار همگرایی k پس از کاهش بازه نویز در on policy IRL

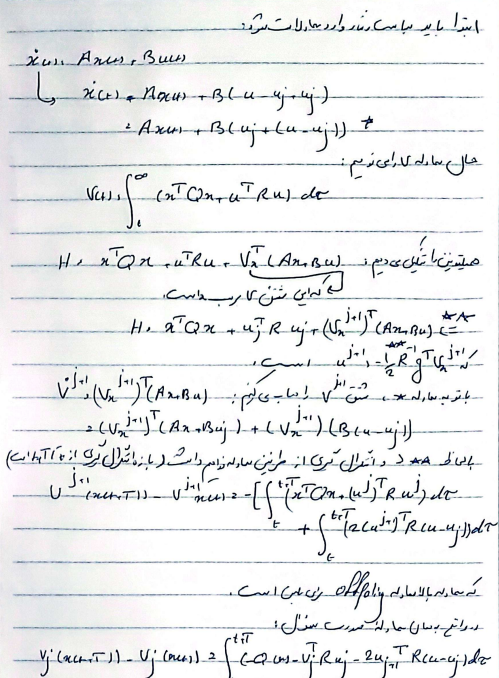
با اعمال K حاصل از IRL رفتار متغیر‌های حالت را با کد زیر رسم می‌کنیم. (نیاز به توضیح اضافه نیست صرفا شبیه سازی سیستم است)

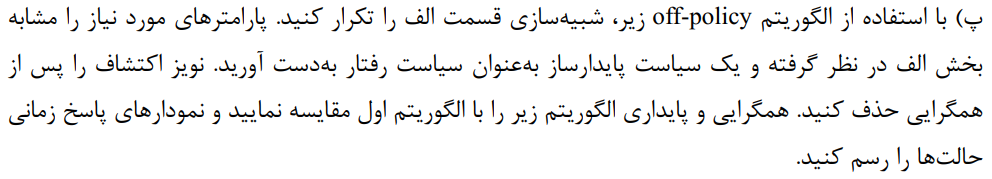


خروجی به صورت زیر است:

شکل 9: متغیرهای حالت پس از اعمال سیاست کنترلی بهینه

**صورت سوال بخش ب:**

**پاسخ بخش ب:**

**صورت سوال بخش پ:**

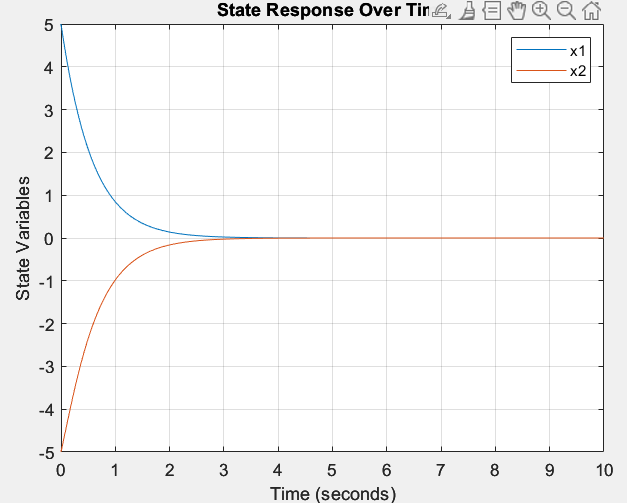
**پاسخ بخش پ:**

در این بخش نیز، کد مشابه قبل است و صرفا محاسبه پاداش به صورت زیر تغییر یافته است:



K بهینه به صورت زیر بدست می‌آید:

که بسیار نزدیک به بهره بدست آمده با دستور LQR است.

حالات سیستم نیز به صورت زیر بدست می‌آیند.

شکل 10: نمودار حالات سیستم

همگرایی با الگوریتم Off Policy دیر تر از On Policy رخ می‌دهد، اما پاسخ Off Policy با حالت بهینه منطبق‌تر است، هرچند که پاسخ حالت On Policy نیز تا حد خوبی تطابق دارد.

1. از دوره‌ی یادگیری تقویتی David Silver [↑](#footnote-ref-1)