



دانشگاه صنعتی شریف  
دانشکده مهندسی هوافضا

پروژه کارشناسی ارشد  
مهندسی فضا

عنوان:

## هدایت یادگیری تقویتی مقاوم مبتنی بر بازی دیفرانسیلی در محیط‌های پویایی چندجسمی با پیشران کم

نگارش:

علی بنی اسد

استاد راهنما:

دکتر هادی نوبهاری

۱۴۰۳ دی

مَلِكُ الْأَنْفُلِ

به نام خدا

دانشگاه صنعتی شریف

دانشکده‌ی مهندسی هوا فضا

پروژه کارشناسی ارشد

عنوان: هدایت یادگیری تقویتی مقاوم مبتنی بر بازی دیفرانسیلی در محیط‌های پویای چندجسمی  
با پیشران کم

نگارش: علی بنی اسد

کمیته‌ی ممتحنین

استاد راهنمای: دکتر هادی نوبهاری      امضاء:

استاد مشاور: استاد مشاور      امضاء:

استاد مدعو: استاد ممتحن      امضاء:

تاریخ:

## سپاس

از استاد بزرگوارم جناب آقای دکتر نوبهاری که با کمک‌ها و راهنمایی‌های بی‌دriegشان، بنده را در انجام این پروژه یاری داده‌اند، تشکر و قدردانی می‌کنم. از پدر دلسوزم ممنونم که در انجام این پروژه مرا یاری نمود. در نهایت در کمال تواضع، با تمام وجود بر دستان مادرم بوسه می‌زنم که اگر حمایت بی‌دriegش، نگاه مهربانش و دستان گرمش نبود برگ این دست نوشته و پروژه وجود نداشت.

## چکیده

در این پژوهش، از یک روش مبتنی بر نظریه بازی<sup>۱</sup> به منظور کنترل وضعیت استند سه درجه آزادی چهارپره استفاده شده است. در این روش بازیکن اول سعی در ردگیری ورودی مطلوب می‌کند و بازیکن دوم با ایجاد اغتشاش سعی در ایجاد خطا در ردگیری بازیکن اول می‌کند. در این روش انتخاب حرکت با استفاده از تعادل نش<sup>۲</sup> که با فرض بدترین حرکت دیگر بازیکن است، انجام می‌شود. این روش نسبت به اغتشاش ورودی و همچنین نسبت به عدم قطعیت مدل‌سازی می‌تواند مقاوم باشد. برای ارزیابی عملکرد این روش ابتدا شبیه‌سازی‌هایی در محیط سیمولینک انجام شده است و سپس، با پیاده‌سازی روی استند سه درجه آزادی صحت عملکرد کنترل‌کننده تایید شده است.

**کلیدواژه‌ها:** چهارپره، بازی دیفرانسیلی، نظریه بازی، تعادل نش، استند سه درجه آزادی، مدل‌بنا، تنظیم‌کننده مربعی خطی

---

<sup>1</sup>Game Theory

<sup>2</sup>Nash Equilibrium

# فهرست مطالب

۱	۱	مقدمه
۱	۱-۱	انگیزه پژوهش
۲	۲-۱	تعریف مسئله
۳	۳-۱	یادگیری تقویتی
۴	۴-۱	یادگیری تقویتی چندعاملی
۴	۵-۱	محتوای گزارش
۵	۲	پیشینه پژوهش
۵	۱-۲	ماموریت‌های بین مداری
۷	۲-۲	یادگیری تقویتی
۸	۳-۲	پیشینه‌ی پژوهش یادگیری تقویتی چندعاملی
۱۰	۳	مدل‌سازی محیط یادگیری سه جسمی
۱۰	۱-۳	مسئله‌ی سه‌جسمی محدود دایره‌ای (CRTBP)
۱۲	۱-۱-۳	۱- لاگرانژ و معادلات حرکت
۱۲	۲-۳	۲- نقاط تعادل لاگرانژ
۱۵	۴	یادگیری تقویتی
۱۵	۱-۴	مفاهیم اولیه

۱۶	.....	۱-۱-۴
۱۶	.....	۲-۱-۴
۱۶	.....	۳-۱-۴
۱۷	.....	۴-۱-۴
۱۷	.....	۵-۱-۴
۱۸	.....	۶-۱-۴
۱۹	.....	۷-۱-۴
۲۰	.....	۸-۱-۴
۲۱	.....	۲-۴
۲۱	.....	۱-۲-۴
۲۲	.....	۲-۲-۴
۲۳	.....	۳-۲-۴
۲۳	.....	۴-۲-۴
۲۵	.....	۳-۴
۲۶	.....	۱-۳-۴
۲۶	.....	۲-۳-۴
۲۸	.....	۴-۴
۲۸	.....	۱-۴-۴
۲۸	.....	۲-۴-۴
۲۹	.....	۳-۴-۴
۲۹	.....	۴-۴-۴
۲۹	.....	۵-۴-۴
۳۰	.....	۶-۴-۴
۳۰	.....	۷-۴-۴

۳۱	.....	۸-۴-۴ اكتشاف و بهره‌برداری در SAC
۳۲	.....	۹-۴-۴ شبکه SAC
۳۳	.....	۵-۴ عامل بهینه‌سازی سیاست مجاور
۳۴	.....	۱-۵-۴ سیاست در الگوریتم PPO
۳۵	.....	۲-۵-۴ اكتشاف و بهره‌برداری در PPO
۳۵	.....	۳-۵-۴ شبکه PPO
۳۷	.....	۵ شبیه‌سازی عامل در محیط سه جسمی
۳۷	.....	۱-۵ طراحی عامل
۳۷	.....	۱-۱-۵ فضای حالت
۳۸	.....	۲-۱-۵ فضای عمل
۳۹	.....	۳-۱-۵ تابع پاداش
۴۰	.....	۲-۵ شبیه‌سازی عامل
۴۰	.....	۱-۲-۵ پارامترهای یادگیری الگوریتم‌های مورد استفاده
۴۳	.....	۲-۲-۵ فرآیند آموزش
۴۵	.....	۶ یادگیری تقویتی چندعاملی
۴۵	.....	۱-۶ تعاریف و مفاهیم اساسی
۴۷	.....	۲-۶ نظریه بازی‌ها
۴۷	.....	۱-۲-۶ تعادل نش
۴۸	.....	۲-۲-۶ بازی مجموع صفر
۴۹	.....	۳-۶ گرادیان سیاست عمیق قطعی دوعلاملی
۴۹	.....	۱-۳-۶ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی
۴۹	.....	۲-۳-۶ معماری MA-DDPG در بازی‌های مجموع صفر
۵۰	.....	۳-۳-۶ آموزش MA-DDPG در بازی‌های مجموع صفر

۵۱	.....	۴-۳-۶ اکتشاف در MA-DDPG
۵۱	.....	۵-۳-۶ شبکه MA-DDPG برای بازی‌های دوعلاملی مجموع صفر
۵۳	.....	۶-۳-۶ مزایای MA-DDPG در بازی‌های مجموع صفر
۵۳	.....	۴-۶ عامل گرادیان سیاست عمیق قطعی تا خیری دوگانه چندعاملی
۵۳	... MATD3	۱-۴-۶ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه حل
۵۴	.....	۲-۴-۶ معماری MATD3 در بازی‌های مجموع صفر
۵۴	.....	۳-۴-۶ آموزش MATD3
۵۵	.....	۴-۴-۶ اکتشاف در MATD3
۵۶	.....	۵-۴-۶ شبکه MATD3 برای بازی‌های چندعاملی مجموع صفر
۵۸	.....	۶-۴-۶ مزایای MATD3 در بازی‌های مجموع صفر
۵۸	.....	۵-۶ عامل عملگر نقاد نرم چندعاملی
۵۸	... MASAC	۱-۵-۶ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه حل
۵۹	.....	۲-۵-۶ معماری MASAC در بازی‌های مجموع صفر
۵۹	.....	۳-۵-۶ آموزش MASAC
۶۱	.....	۴-۵-۶ اکتشاف در MASAC
۶۱	.....	۵-۵-۶ شبکه MASAC برای بازی‌های چندعاملی مجموع صفر
۶۳	.....	۶-۵-۶ مزایای MASAC در بازی‌های مجموع صفر
۶۳	.....	۶-۶ عامل بهینه‌سازی سیاست مجاور چندعاملی
۶۳	... MAPPO	۱-۶-۶ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه حل
۶۴	.....	۲-۶-۶ معماری MAPPO در بازی‌های مجموع صفر
۶۴	.....	۳-۶-۶ آموزش MAPPO
۶۶	.....	۴-۶-۶ اکتشاف در MAPPO
۶۶	.....	۵-۶-۶ شبکه MAPPO برای بازی‌های چندعاملی مجموع صفر
۶۷	.....	۶-۶-۶ مزایای MAPPO در بازی‌های مجموع صفر

۶۹	۷	ارزیابی عملکرد سخت افزار در حلقه عامل
۷۰	۱-۷	پیکربندی آزمایشگاهی
۷۱	۱-۱-۷	رایانه‌ی میزبان (Host PC)
۷۱	۲-۱-۷	رایانه‌ی هدف (Target Board)
۷۱	۳-۱-۷	شبکه و پروتکل‌های ارتباطی
۷۱	۴-۱-۷	شبکه و پروتکل‌های ارتباطی
۷۲	۲-۷	لایه‌ی نرم افزاری مبتنی بر ROS 2
۷۳	۱-۲-۷	معماری گره‌ها
۷۳	۲-۲-۷	رابطه‌ای واسط داده
۷۳	۳-۲-۷	پیکربندی بلادرنگ و زمان‌بندی
۷۴	۴-۲-۷	ثبت و مانیتورینگ داده
۷۴	۳-۷	بودجه‌ی زمان واقعی و تحلیل تأخیر
۷۵	۴-۷	بهینه‌سازی و استقرار مدل یادگیری
۷۵	۱-۴-۷	تبدیل قالب PyTorch به ONNX
۷۶	۲-۴-۷	کوانتیزاسیون عددی INT8
۷۶	۵-۷	سناریوهای اعتبارسنجی
۷۶	۱-۵-۷	شرایط اولیه تصادفی
۷۷	۲-۵-۷	اغتشاش عملگر
۷۷	۳-۵-۷	عدم تطابق مدل
۷۷	۴-۵-۷	مشاهده‌ی ناقص
۷۷	۵-۵-۷	نویز حسگر
۷۸	۶-۵-۷	تأخیر زمانی
۷۸	۶-۷	جمع‌بندی
۷۹	۸	ارزیابی و نتایج یادگیری

۷۹	۱-۸ تنظیمات آزمایشی
۸۰	۲-۸ مقایسه مسیرها و فرمان پیشران
۸۰	۱-۲-۸ الگوریتم DDPG
۸۱	۲-۲-۸ الگوریتم PPO
۸۲	۳-۲-۸ الگوریتم SAC
۸۴	۴-۲-۸ الگوریتم TD3
۸۵	۳-۸ ارزیابی مقاومت الگوریتم‌ها
۸۵	۱-۳-۸ مقایسه الگوریتم‌های تک‌عاملی و چند‌عاملی DDPG
۸۶	۲-۳-۸ مقایسه الگوریتم‌های تک‌عاملی و چند‌عاملی PPO
۸۷	۳-۳-۸ مقایسه الگوریتم‌های تک‌عاملی و چند‌عاملی SAC
۸۸	۴-۳-۸ مقایسه الگوریتم‌های تک‌عاملی و چند‌عاملی TD3
۸۸	۴-۸ مقایسه جامع الگوریتم‌ها
۸۹	۱-۴-۸ مقایسه الگوریتم‌های تک‌عاملی
۹۰	۲-۴-۸ مقایسه الگوریتم‌های چند‌عاملی
۹۰	۵-۸ تحلیل پایداری و همگرایی
۹۱	۶-۸ مقایسه با معیارهای مرجع

# فهرست جداول

۱-۳	مقادیر عددی برای مسئله سه‌جسمی محدود (سیستم زمین-ماه)	۱۱
۲-۳	مقادیر عددی نقاط لاغرانژ برای مسئله سه‌جسمی محدود سیستم زمین-ماه	۱۴
۱-۵	ویژگی‌های الگوریتم‌های مورد استفاده در شبیه‌سازی	۴۰
۲-۵	جدول پارامترها و مقادیر پیش‌فرض الگوریتم DDPG [۱]	۴۲
۳-۵	جدول پارامترها و مقادیر پیش‌فرض الگوریتم TD3 [۱]	۴۲
۴-۵	جدول پارامترها و مقادیر پیش‌فرض الگوریتم SAC [۱]	۴۳
۵-۵	جدول پارامترها و مقادیر پیش‌فرض الگوریتم PPO [۱]	۴۳
۱-۷	گره‌های اصلی سامانه و وظایف آن‌ها	۷۳
۲-۷	بودجه‌ی زمانی چرخه‌ی کنترل در بسامد Hz 100	۷۵
۳-۷	تأثیر کوانتیزاسیون بر اندازه‌ی مدل و تأخیر استنتاج	۷۶

# فهرست تصاویر

۱۱	هندسه مسئله سه بدن محدود	۱-۳
۱۳	نقاط لاگرانژ	۲-۳
۱۶	حلقه تعامل عامل و محیط	۱-۴
۴۱	ساختار شبکه عصبی عامل	۱-۵
۴۱	ساختار شبکه عصبی نقاد	۲-۵
۴۶	حلقه تعامل عامل‌های یادگیری تقویتی چند عاملی با محیط	۱-۶
۷۰	شماتیک سخت‌افزار در حلقة	۱-۷
۸۰	مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چند‌عاملی DDPG. مشاهده می‌شود که نسخه بازی مجموع صفر مسیر مستقیم‌تری را با انحراف کمتر از مسیر بهینه طی می‌کند.	۱-۸
۸۱	مقایسه مسیر و فرمان پیش‌ران دو الگوریتم تک‌عاملی و چند‌عاملی DDPG. نمودارهای پایین نشان‌دهنده فرمان پیش‌ران در طول زمان است که در نسخه بازی مجموع صفر، الگوی منظم‌تری را نشان می‌دهد و اوج‌های پیش‌ران کمتری دارد.	۲-۸
۸۲	مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چند‌عاملی PPO. نسخه بازی مجموع صفر همگرایی بهتری به مسیر هدف را نشان می‌دهد، به خصوص در مراحل نزدیک شدن به هدف.	۳-۸
۸۲	مقایسه مسیر و فرمان پیش‌ران دو الگوریتم تک‌عاملی و چند‌عاملی PPO. فرمان‌های پیش‌ران در نسخه بازی مجموع صفر از نظر توزیع انرژی متوازن‌تر است و نوسانات کمتری را نشان می‌دهد.	۴-۸

- ۵-۸ مقایسه مسیر طی شده در دو الگوریتم تک عاملی و چند عاملی SAC. مسیرهای تولید شده توسط هر دو نسخه از کیفیت بالایی برخوردارند، اما نسخه بازی مجموع صفر در مناطق با گرادیان جاذبه پیچیده عملکرد پایدارتری را نشان می‌دهد. . . . .

۶-۸ مقایسه مسیر و فرمان پیشran دو الگوریتم تک عاملی و چند عاملی SAC. نسخه بازی مجموع صفر مصرف سوخت متعادل‌تری را در طول مسیر نشان می‌دهد که می‌تواند منجر به صرفه‌جویی در منابع شود. . . . .

۷-۸ مقایسه مسیر طی شده در دو الگوریتم تک عاملی و چند عاملی TD3. مسیرهای تولید شده توسط نسخه بازی مجموع صفر نشان‌دهنده کاهش انحراف از مسیر بهینه و همگرایی سریع‌تر به هدف است. . . . .

۸-۸ مقایسه مسیر و فرمان پیشran دو الگوریتم تک عاملی و چند عاملی TD3. فرمان‌های پیشran در نسخه بازی مجموع صفر از توزیع یکنواخت‌تری برخوردار است که نشان‌دهنده استفاده بهینه‌تر از منابع پیشranش می‌باشد. . . . .

۹-۸ مقایسه مجموع پاداش دو الگوریتم تک عاملی و چند عاملی DDPG در سناریوهای مختلف. نسخه بازی مجموع صفر در اکثر سناریوهای خود را به خصوص در شرایط اغتشاش در عملگرها و عدم تطابق مدل، عملکرد بهتری را نشان می‌دهد. . . . .

۱۰-۸ مقایسه مجموع پاداش دو الگوریتم تک عاملی و چند عاملی PPO در سناریوهای مختلف. نسخه بازی مجموع صفر در سناریوهای تأخیر زمانی و نویز حسگر برتری قابل توجهی نشان می‌دهد. . . . .

۱۱-۸ مقایسه مجموع پاداش دو الگوریتم تک عاملی و چند عاملی SAC در سناریوهای مختلف. هر دو نسخه عملکرد نسبتاً خوبی دارند، اما نسخه بازی مجموع صفر در شرایط عدم تطابق مدل و مشاهده ناقص برتری بیشتری نشان می‌دهد. . . . .

۱۲-۸ مقایسه مجموع پاداش دو الگوریتم تک عاملی و چند عاملی TD3 در سناریوهای مختلف. نسخه بازی مجموع صفر در تمام سناریوهای عملکرد بهتری را نشان می‌دهد، با برتری قابل توجه در سناریوهای اغتشاش در عملگرها و نویز حسگر. . . . .

۱۳-۸ مقایسه مجموع پاداش الگوریتم‌های تک عاملی در سناریوهای مختلف. در اکثر سناریوهای SAC و TD3 عملکرد بهتری نسبت به DDPG و PPO نشان می‌دهند، با عنوان برترین الگوریتم در شرایط نویز حسگر و تأخیر زمانی. . . . .

۱۴-۸ مقایسه مجموع پاداش الگوریتم‌های چندعاملی در سناریوهای مختلف. TD3 مبتنی بر بازی

مجموع صفر در اکثر سناریوها بهترین عملکرد را دارد، در حالی که SAC در سناریوهای نویز

حسگر و تأخیر زمانی برتری نشان می‌دهد. . . . .

۹۰

# فهرست الگوریتم‌ها

۲۴	گرادیان سیاست عمیق قطعی	۱
۲۷	عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه	۲
۳۲	عامل عملگرد نقاد نرم	۳
۳۶	بهینه‌سازی سیاست مجاور (PPO-Clip)	۴
۵۲	گرادیان سیاست عمیق قطعی چندعاملی برای بازی‌های مجموع صفر	۵
۵۷	عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه دوعلاملی	۶
۶۲	عامل عملگر نقاد نرم دوعلاملی	۷
۶۷	عامل بهینه‌سازی سیاست مجاور دوعلاملی	۸

# فصل ۱

## مقدمه

در سال‌های آغازین عصر فضا، فرآیند هدایت فضایی‌ها عمدهاً بر مبانی دینامیک کلاسیک و کنترل خطی استوار بوده است. با این حال، پیچیدگی روزافزون مأموریت‌های کنونی مانند سفرهای میان‌سیاره‌ای با پیشرانکم و شبکه‌های انبوه ماهواره‌ای در مدار زمین، موجب دوچندان شدن ضرورت بهره‌گیری از روش‌های هوشمند و تطبیق‌پذیر شده‌است.

## ۱-۱ انگیزه پژوهش

در دو دهه‌ی اخیر، مأموریت‌های فضایی به دلیل کوچک‌سازی سامانه‌ها، توسعه‌ی وسائل الکترونیک مقرن به صرفه و افزایش ظرفیت‌های پرتابی، تحولات بنیادینی را تجربه کرده است. از پروژه‌های علمی بین سیاره‌ایی گفته تا منظومه‌های انبوه ماهواره‌ایی در مدارهای پایین زمین، همگی با چالش فراگیر هدایت بهینه در حضور عدم قطعیت‌ها مواجه‌اند. در مسیرهای فرا-قمری<sup>۱</sup> و به طور خاص در ناحیه‌های ناپایدار نقاط لاغرانژ در چارچوب مسئله‌ی سه جسمی کروی محدود و دایروی<sup>۲</sup>، طراحی سامانه‌ی کنترل مستلزم توانایی تضمین همزمان پایداری ایستا و بهره‌وری سوخت با نیروی پیشرانکم<sup>۳</sup> است.

همراستا با این تحولات، ظهور و گسترش الگوریتم‌های یادگیری تقویتی عمیق<sup>۴</sup>، امکانات نوینی را برای طراحی کنترل‌کننده‌های تطبیقی فراهم آورده است. با این حال، غالب رویکردهای رایج بر سناریوهای تک‌عاملی و اتکا به مدل‌های دینامیکی دقیق استوارند. غیاب یک استراتژی مقاوم در مواجهه با اغتشاشات مدل و تغییرات

<sup>1</sup>Trans-lunar

<sup>2</sup>Circular Restricted Three-Body Problem (CRTBP)

<sup>3</sup>Low-thrust

<sup>4</sup>Deep Reinforcement Learning (DRL)

محیطی از جمله خطای تراست در پیشran و تأخیر در سیگنال‌های حسگر منجر به فاصله‌ی چشمگیر عملکرد واقعی از پیش‌بینی‌های حاصل از شبیه‌سازی‌های ایده‌آل می‌گردد. این پژوهش بر آن است تا گسیست اشاره‌شده را با بهره‌گیری از چارچوب یادگیری تقویتی چندعاملی مقاوم مرتفع سازد و بدین وسیله، اطمینان هدایت پیشran کم در مسئله‌ی سه‌جسمی محدود دایره‌ای<sup>۵</sup> را افزایش دهد.

## ۲-۱ تعریف مسئله

در سال‌های اخیر، پیشرفت‌های فناوری در زمینه‌های مختلف، از جمله کنترل پرواز، پردازش سیگنال و هوش مصنوعی، به افزایش کاربردهای ماهواره با پیشran کم در منظمه زمین-ماه کمک کرده است. ماهواره با پیشran کم می‌تواند برای تعقیب ماهواره‌ها، انتقال مداری و استقرار ماهواره‌ها استفاده شود. روش‌های هدایت بهینه قدیمی جهت کنترل ماهواره‌ها اغلب نیازمند فرضیات ساده‌کننده، منابع محاسباتی فراوان و شرایط اولیه مناسب هستند. الگوریتم‌های مبتنی بر یادگیری تقویتی این توانایی را دارند که بدون مشکلات اشاره‌شده هدایت ماهواره را انجام دهند. به دلیل ساختار شبکه‌ای، این الگوریتم‌ها می‌توانند امکان محاسبات درونی<sup>۶</sup> را فراهم کنند.

هدف از این پژوهش، طراحی سیاست کنترلی برای یک فضایپیما به جرم  $m$  است که در میدان جاذبه‌ی سیستم زمین-ماه به صورت دو بعدی مدل شده است. ویژگی‌های این سامانه در ادامه آورده شده است.

- پویایی‌ها: معادلات حرکت در چارچوب مرجع چرخان به صورت مجموعه غیرخطی  $u$  ( $\dot{x} = f(x) + g(x)$ ) است که در میدان جاذبه‌ی نوشته می‌شود که  $x = [x, y, \dot{x}, \dot{y}]^\top$  بردار حالت و  $u$  بردار تراست با کران  $u_{\max} \leq u \leq u$  است.

- عدم قطعیت‌ها: عوامل عدم قطعیت شامل شرایط اولیه تصادفی، اغتشاش در عملگرها، عدم تطابق مدل، مشاهده ناقص، نویز حسگر و تأخیر زمانی هستند که همگی بر عملکرد و پایداری سیستم تأثیر می‌گذارند.

- صورت بازی دیفرانسیلی: فضایپیما و طبیعت (اغتشاشات) به ترتیب به عنوان عامل کنترلی و حریف مزاحم مدل شده است. مسئله به عنوان یک بازی مجموعه‌صفر<sup>۷</sup> در افق زمان محدود  $t_f$  فرمول بندی شده‌اند.

صورت کامل مسئله را می‌توان با یافتن سیاست  $u^*$  تعریف کرد که معیار بهینه‌سازی هزینه‌ی تجمعی است.

---

<sup>5</sup>CRTBP

<sup>6</sup>On-board Computing

<sup>7</sup>Zero-Sum

## ۳-۱ یادگیری تقویتی

یادگیری تقویتی<sup>۸</sup> شاخه‌ای از یادگیری ماشین است که در آن یک عامل از طریق تعامل پیاپی با محیط می‌آموزد چه توالی اقدام‌هایی  $a_t \in \mathcal{A}$  را انتخاب کند تا بازده تجمعی آینده را بیشینه کند. یک فرایند تصمیم‌گیری مارکوف<sup>۹</sup> به صورت  $\langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$  تعریف می‌شود که در آن

- $\mathcal{S}$ : مجموعه حالات،

- $p(s'|s, a)$ : دینامیک انتقال،

- $r(s, a)$ : پاداش آنی،

- $\gamma \in [0, 1]$ : ضریب تنزیل.

سیاست<sup>۱۰</sup>  $\pi(a|s)$  احتمال انتخاب اقدام  $a$  در وضعیت  $s$  را بیان می‌کند. هدف بیشینه‌سازی برگشت<sup>۱۱</sup>

$$G_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k} \quad (1-1)$$

است. روش‌های RL معمولاً در دو دسته‌ی ارزش‌محور (مانند Q-learning و DQN) و سیاست‌محور (مانند REINFORCE) جای می‌گیرند؛ ترکیب این دو به چارچوب Actor–Critic منتهی می‌شود که در آن یک بازیکن (Actor) سیاست را بهروزرسانی می‌کند و یک منتقد (Critic) ارزش یا  $Q$  را برآورد می‌نماید [۲].

در حضور فضاهای پیوسته‌ی حالت–عمل، الگوریتم‌های گرادیان سیاست عمیق مانند TD3، DDPG و PPO با تکیه بر شبکه‌های عصبی به عنوان تقریب‌گر توابع، کارایی بالایی نشان داده‌اند. در این پژوهش، خانواده‌ی Actor–Critic پایه‌ی توسعه‌ی کنترل‌کننده پیشنهاد شده‌است.

## ۴-۱ یادگیری تقویتی چندعاملی

در یادگیری تقویتی چندعاملی<sup>۱۲</sup>، فضای تصمیم‌گیری به صورت یک بازی مارکفی<sup>۱۳</sup> با مجموعه عامل‌ها  $= \mathcal{N}$  مدل شده‌است که در آن هر عامل سیاست  $\pi_i$  مختص خود را با هدف بیشینه‌سازی پاداش تجمعی

<sup>8</sup>Reinforcement Learning (RL)

<sup>9</sup>Markov Decision Process (MDP)

<sup>10</sup>Policy

<sup>11</sup>Return

<sup>12</sup>Multi-Agent Reinforcement Learning (MARL)

<sup>13</sup>Markov Games (MG)

کسب می‌کند. در سناریوهای رقابتی دونفره، این پژوهش از چارچوب بازی‌های Zero-Sum استفاده شده و مفهوم تعادل نش<sup>۱۴</sup> به عنوان معیار پایداری سیاست مطرح شده است.

رویکرد آموزش متمرکز، اجرا توزیع شده<sup>۱۵</sup> با جدا کردن مرحله‌ی آموزش که در آن اطلاعات خصوصی همه عامل‌ها در دسترس است از اجرا که در آن هر عامل صرفاً بر مشاهده‌ی محلی اتکا می‌کند، تعادل بین کارایی و مقیاس‌پذیری را برقرار می‌سازد. این معماری مخصوصاً در حضور تعامل‌های ضعیف عامل‌ها مفید است، زیرا هزینه‌ی ارتباطی در زمان اجرا را حذف می‌کند.

## ۱-۵ محتوای گزارش

---

<sup>14</sup>Nash Equilibrium

<sup>15</sup>Centralized Training with Decentralized Execution (CTDE)

## فصل ۲

### پیشینه پژوهش

#### ۱-۲ ماموریت‌های بین مداری

هدایت فضایپیماها معمولاً با استفاده از ایستگاه‌های زمینی انجام می‌شود. با این حال، این تکنیک‌ها دارای محدودیت‌هایی از جمله حساسیت به قطع ارتباطات، تاخیرهای زمانی و محدودیت‌های منابع محاسباتی هستند. الگوریتم‌های یادگیری تقویتی و بازی‌های دیفرانسیلی می‌توانند برای بهبود قابلیت‌های هدایت فضایپیماها، از جمله مقاومت در برابر تغییرات محیطی، کاهش تاخیرهای ناشی از ارتباطات زمینی و افزایش کارایی محاسباتی، مورد استفاده قرار گیرند.

هدایت فضایپیماها معمولاً پیش از پرواز انجام می‌شود. این روش‌ها می‌توانند از تکنیک‌های بهینه‌سازی فرآگیر [۳] یا برنامه‌نویسی غیرخطی برای تولید مسیرها و فرمان‌های کنترلی بهینه استفاده کنند. با این حال، این روش‌ها معمولاً حجم محاسباتی زیادی دارند و برای استفاده درون‌سفینه‌ای نامناسب هستند [۴]. یادگیری ماشین می‌تواند برای بهبود قابلیت‌های هدایت فضایپیماها استفاده شود. کنترل‌کننده شبکه عصبی حلقه‌بسته می‌تواند برای محاسبه سریع و خودکار تاریخچه کنترل استفاده شود. یادگیری تقویتی نیز می‌تواند برای یادگیری رفتارهای هدایت بهینه استفاده شود.

روش‌های هدایت و بهینه‌سازی مسیر فضایپیماها به‌طور کلی به راه حل‌های اولیه مناسب نیاز دارند. در مسائل چند جسمی، طراحان مسیر اغلب حدسهای اولیه کم‌هزینه‌ای برای انتقال‌ها با استفاده از نظریه سیستم‌های دینامیکی و منیفولد‌های ثابت [۵، ۶] ایجاد می‌کنند.

شبکه‌های عصبی ویژگی‌های جذابی برای فعال‌سازی انجام هدایت در فضایپیما دارند. به عنوان مثال، شبکه‌های عصبی می‌توانند به‌طور مستقیم از تخمین‌های وضعیت به دستورهای پیشران‌کنترلی که با محدودیت‌های

مأموریت سازگار است، برسند. عملکرد هدایت شبکه‌های عصبی در مطالعاتی مانند فرود بر سیارات [۱۷]، عملیات نزدیکی به سیارات [۸] و کنترل فضاییما با پیشران ازدسترفته [۹] نشان داده شده است. تازه‌ترین پیشرفت‌های تکنیک‌های یادگیری ماشین در مسائل خودکارسازی درونی به طور گسترده‌ای مورد مطالعه قرار گرفته‌اند؛ از پژوهش‌های اولیه تا توانایی‌های پیاده‌سازی. به عنوان مثال، الگوریتم‌های یادگیری ماشین ابتدایی در فضایی‌ها مریخی نورد برای کمک به شناسایی ویژگی‌های زمین‌شناسی تعییه شده‌اند. الگوریتم AEGIS Curiosity Opportunity و Spirit را فعال دارد [۱۰]. در کامپیوتر پرواز اصلی، فرآیند دقت افزایی (Refinement Process) نیاز به ۹۶ تا ۹۴ ثانیه دارد [۱۱]، که به طور قابل توجهی کمتر از زمان نیاز برای ارسال تصاویر به زمین و انتظار برای انتخاب دستی توسط دانشمندان است. برنامه‌های آینده برای کاربردهای یادگیری ماشین درون‌سفینه شامل توانایی‌های رباتیکی درون‌سفینه برای فضایی‌ها Perseverance [۱۲] و شناسایی عیوب برای Europa Clipper [۱۴] می‌شود. الگوریتم‌های یادگیری ماشین دارای پتانسیلی انجام سهم مهمی در مأموریت‌های اتوماسیون آینده دارند.

علاوه بر رباتیک سیاره‌ای، پژوهش‌های مختلفی به استفاده از تکنیک‌های مختلف یادگیری ماشین در مسائل نجومی پرداخته‌اند. در طراحی مسیر عملکرد رگرسیون معمولاً مؤثرتر هست. به عنوان مثال، از یک شبکه عصبی (NN) در بهینه‌سازی مسیرهای رانشگر کم‌پیشran استفاده شده است [۱۵]. پژوهش‌های جدید شامل شناسایی انتقال‌های هتروکلینیک [۱۶]، اصلاح مسیر رانشگر کم‌پیشran [۱۷] و تجزیه و تحلیل مشکلات ازدست‌رفتن رانشگر [۹] می‌شود.

تکنیک‌های یادگیری نظری می‌توانند نتایج مطلوبی تولید کنند؛ اما، دارای محدودیت‌های قابل توجهی هستند. یکی از این محدودیت‌ها این است که این رویکردها بر وجود دانش پیش از فرآیند تصمیم‌گیری متکی هستند. این امر مستلزم دقیق‌بودن داده‌های تولیدشده توسط کاربر برای نتایج مطلوب و همچنین وجود تکنیک‌های موجود برای حل مشکل کنونی و تولید داده است.

در سال‌های اخیر، قابلیت یادگیری تقویتی (RL) در دستیابی به عملکرد بهینه در بخش‌هایی با ابهام محیطی قابل توجه، به اثبات رسیده است [۱۸، ۱۹]. هدایت انجام‌شده توسط RL را می‌توان به صورت گسترده بر اساس فاز پرواز دسته‌بندی کرد. مسائل فرود [۲۰، ۲۱] و عملیات در نزدیکی اجسام کوچک [۸، ۷]، از حوزه‌های پژوهشی هستند که از RL استفاده می‌کنند. تحقیقات دیگر شامل مواجهه تداخل خارجی جوی [۲۲]، نگهداری ایستگاهی [۲۳] و هدایت به صورت جلوگیری از شناسایی [۲۴] است. مطالعاتی که فضایی‌ها رانشگر کم‌پیشran را در یک چارچوب دینامیکی چند بدنه با استفاده از RL انجام‌شده است، شامل طراحی انتقال با استفاده از Q-learning [۲۵] و هدایت نزدیکی مدار [۲۷] است.

## ۲-۲ یادگیری تقویتی

از نخستین صورت‌بندی‌های فرایند تصمیم‌گیری مارکفی در یادگیری تقویتی، پژوهش بر آن بوده است که عامل بتواند با اجرای عمل‌ها و دریافت پاداش، سیاستی برای بیشینه‌سازی برگشت بیاموزد. تبیین جامع این چارچوب و الگوریتم‌های بنیادین در ویرایش دوم کتاب سوتون و بارتون بهمثابه مرجع کلاسیک این حوزه ارائه شده و همچنان مبنای بسیاری از آثار معاصر است [۲].

دهه‌ی ۱۹۹۰ ملادی شاهد شکل‌گیری روش‌هایی بر پایه‌ی ارزش<sup>۱</sup> نظری Q-learning و نخستین رویکردهای گرادیان سیاست بود؛ با وجود این، محدودیت توان محاسباتی و فقدان داده‌ی فراوان، سرعت رشد را کند می‌کرد. ورود شبکه‌های عصبی عمیق نقطه‌ی عطفی بود: مقاله‌ی معروف دیپ‌مایندر<sup>۲</sup> نشان داد که شبکه‌ی Q عمیق (DQN) می‌تواند صرفاً از پیکسل‌های بازی آتاری سیاستی نزدیک به انسان بیاموزد [۲۸].

موفقیت DQN نگاه‌ها را به‌سوی گرادیان سیاست مقیاس‌پذیر معطوف ساخت. بهینه‌سازی ناحیه‌ی اطمینان<sup>۳</sup> تضمین بهبود یکنواخت سیاست را فراهم کرد [۲۹]، و روش A3C با موازنی‌سازی بازیگران، سرعت یادگیری را چند برابر افزایش داد [۳۰]. کمی بعد، DDPG اولین بار گرادیان سیاست قطعی را به فضاهای عمل پیوسته وارد کرد [۳۱]. سپس PPO با ساده‌سازی قیود TRPO و کاهش پارامترهای حساس، به انتخاب پیش‌فرض بسیاری از کاربردهای مهندسی بدل شد [۳۲].

با گسترش دامنه‌ی مسائل، پایداری و کارایی داده به چالش اصلی بدل گشت. TD3 نشان داد که کمینه‌کردن میان دو منتقد می‌تواند برآورد بیش از حد Q را مهار کند [۳۳]، و SAC با افزودن بند آنتروپی، هم‌زمان اکتشاف و بازده را بهبود داد [۳۴].

در محیط‌های پرخطر یا گران، جمع‌آوری داده‌ی برخط ناممکن است؛ از این‌رو RL آفلاین مطرح شد. روش CQL با برقراری کران محافظه‌کارانه بر Q-value از گرایش خارج از توزیع جلوگیری می‌کند [۳۵]، و مرور اخیر پراودنسیو و همکاران طبقه‌بندی جامعی از چالش‌های باز این حوزه ارائه داده است [۳۶].

هم‌زمان، دغدغه‌ی ایمنی و تبیین در سامانه‌های واقعی پرنگ شد. مرور سال ۲۰۲۲ نشان می‌دهد که ترکیب قیدهای سخت، توابع جریمه‌ی ریسک و شبیه‌سازی محیط‌های بدینانه سه خط اصلی ایمنی در RL هستند [۳۷]. سلسه‌مراتب نیز با هدف انتقال دانش و تسریع یادگیری مورد توجه قرار گرفت و یک مطالعه‌ی جامع در ACM Computing Surveys چهار چالش کشف زیرکار، یادگیری اشتراک‌پذیر، انتقال و مقیاس‌پذیری را بر جسته می‌کند [۳۸].

وقتی چند عامل به‌طور هم‌زمان یاد می‌گیرند، پویایی محیط از دید هر عامل غیرایستا می‌شود. مرور جامع

<sup>1</sup>Value

<sup>2</sup>DeepMind

<sup>3</sup>Trust Region Policy Optimization (TRPO)

۲۰۲۴ نشان می‌دهد که چارچوب ناظر مت مرکز - بازیگر توزیع شده<sup>۴</sup> راهکاری موثر برای این چالش است و مباحثی چون تخصیص اعتبار جمعی و کشف تعادل را معرفی می‌کند [۳۹].

پیشرفت‌های یادشده در نهایت به دستاوردهای نمادینی چون AlphaGo [۴۰] و AlphaStar [۴۱] انجامیدند که در بازی‌های Go و StarCraft II از انسان پیشی گرفتند، و معماری توزیع شده‌ی IMPALA نشان داد که چگونه می‌توان هزاران شبیه‌ساز را با بروزرسانی وزن‌های مهم ادغام کرد [۴۲].

به رغم این جهش‌ها، سه شکاف اساسی پابرجا مانده است: ۱) تضمین اینمی سخت‌گیرانه در سناریوهای نزدیک‌برخورد، ۲) کاهش وابستگی به داده‌ی پرهزینه یا نایاب از طریق روش‌های مدل‌بنا و آفلاین، و ۳) مقیاس‌پذیری یادگیری چندعاملی برای سامانه‌های رباتیکی یا فضایی‌پیمای چندگانه.

### ۳-۲ پیشینه‌ی پژوهش یادگیری تقویتی چندعاملی

امروز یادگیری تقویتی چندعاملی (MARL) به عنوان بنیاد اصلی سامانه‌های هوشمند مشارکتی شناخته می‌شود؛ مسیری که از آزمون‌های ساده‌ی دو عاملی در دهه‌ی ۱۹۹۰ آغاز شد و اکنون به معماری‌های توزیع شده‌ی در مقیاس هزاران بازیگر رسیده است. این بخش، به بررسی اینکه چگونه ایده‌ی آموزش مت مرکز - اجرای توزیع شده (CTDE) به پاسخ غالب برای چالش‌های غیرایستایی و انفجار بُعدی بدل شد و چه گام‌هایی هنوز برای اینمی، ناهمگونی و مقیاس‌پذیری باقی مانده است.

دهه‌ی ۱۹۹۰ با مقاله‌ی [۴۳] آغاز شد؛ جایی که برای نخستین بار مقایسه‌ی عامل‌های مستقل با عامل‌های همکار انجام شد و سود ارتباط و اشتراک تجربه به صورت تجربی نشان داده شد. در میانه‌ی دهه‌ی بعد، مورور جامع پانایت و لوك [۴۴] چشم‌اندازی از مسائل تخصیص اعتبار و غیرایستایی ترسیم کرد و دو موضوع یادگیری تیمی و یادگیری همزمان را صورت‌بندی نمود. همزمان، بوشونیو و همکاران [۴۵] ادبیات MARL را در قالب اهداف پایداری دینامیک یادگیری و انطباق با رفتار سایر عامل‌ها جمع‌بندی کردند و راه را برای تحلیل‌های بازی‌محور هموار ساختند.

ورود شبکه‌های عمیق در سال‌های ۲۰۱۶ و ۲۰۱۷ نقطه‌ی عطف بعدی بود؛ منتقد مت مرکز - بازیگر توزیع شده در MADDPG [۴۶] نشان داد که می‌توان از حالت سراسری در فاز آموزش بهره برد، اما سیاست نهایی را صرفاً بر اساس مشاهدات محلی اجرا کرد. در همان سال، Value-Decomposition Networks [۴۷] ایده‌ی تجزیه‌ی خطی پاداش را برای تیم‌های کاملاً تعاونی مطرح کرد و راه را برای فاکتوربندی‌های پیش‌رفته گشود.

۲۰۱۸ شاهد جهش مهمی با QMIX<sup>۵</sup> بود؛ این روش با اعمال قید تکنوا<sup>۶</sup> بر ترکیب مقادیر منفرد، هم

<sup>4</sup>Centralized Training with Decentralized Execution (CTDE)

<sup>5</sup>Monotonic

امکان بهینه‌سازی آف‌پالیسی را فراهم کرد و هم تضمین سازگاری سیاست‌های محلی با ارزش مشترک را برقرار ساخت [۴۸].

سال ۲۰۱۹ به گسترش بسترهای آزمایش اختصاص یافت. چالش استاندارد StarCraft Multi-Agent Challenge (SMAC) بر مبنای StarCraft II معرفی شد و معیار مشترکی برای مقایسه‌ی الگوریتم‌ها را مهیا کرد [۴۹]. هم‌زمان، QTRAN [۵۰] نشان داد که می‌توان بدون قید خطی یا تک‌نوا، تابع ارزش مشترک را به فضای قابل تجزیه تبدیل کرد. از سوی دیگر، MAVEN با افزودن متغیر نهفته‌ی مشترک، کاوش هماهنگ و سلسله‌مراتبی را امکان‌پذیر ساخت [۵۱]. نقطه‌ی اوج همان سال، سامانه‌ی AlphaStar بود که نشان داد ترکیب خودبازی و معماری توزیع‌شده می‌تواند به رتبه‌ی استاد بزرگ<sup>۶</sup> انسان برساند [۴۱].

در ۲۰۲۰ مفهوم نقش‌های در حال ظهور با ROMA [۵۲] معرفی شد تا عامل‌ها بر اساس شbahت رفتاری به‌طور خودکار خوشبندی و اشتراک دانش کنند؛ رویکردی که در نقشه‌های پرترکم SMAC برتری محسوسی نشان داد. پژوهش‌های متأ در ۲۰۲۱، از مرور نظری زانگ و بشار [۵۳] تا محک<sup>۷</sup> تطبیقی پاپوادکیس و همکاران [۵۴]، شکاف‌های باقی‌مانده در تضمین همگرایی و مقیاس را فهرست کردند.

آخرین موج مطالعات بر ناهمگونی و اینمی تمرکز دارد. مرور جامع [۵۵] نشان می‌دهد که تفاوت در قابلیت‌ها و اطلاعات عامل‌ها، مسائلی نظیر تخصیص اعتبار و تعادل را پیچیده‌تر می‌سازد و به الگوریتم‌های سازگار با نقش‌های پویا نیاز دارد.

به‌طور خلاصه، مسیر تاریخی MARL از الگوهای مستقل دهه‌ی ۱۹۹۰ به سامانه‌های توزیع‌شده‌ی امروزی، همواره با سه دغدغه‌ی اصلی هدایت شده است: کنترل انفجار بُعدی توابع ارزش، مقابله با غیرایستایی ناشی از یادگیری هم‌زمان، و انتقال مؤثر تجربه میان عامل‌ها. علی‌رغم پیشرفت‌های شتابان، تضمین اینمی سخت‌گیرانه در محیط‌های شکست‌پذیر، مدیریت نقش‌های پویا در تیم‌های ناهمگون و کاهش نیاز به داده‌ی شبیه‌سازی پرهزینه همچنان چالش‌های باز باقی می‌مانند؛ چالش‌هایی که در این پژوهش با رویکرد ترکیبی مدل‌بنا، مقاوم و چندعاملی پیگیری می‌شوند.

<sup>6</sup>Grandmaster

<sup>7</sup>Benchmark

## فصل ۳

### مدل‌سازی محیط یادگیری سه جسمی

مسیرهای فضایی پیشین تحت تأثیر گرانش یک جسم مرکزی (خورشید، زمین یا سیاره‌ای دیگر) شکل می‌گیرند و توسط اجسام سوم تحت تأثیر قرار می‌گیرند. در برخی موارد، مأموریت فضایپیما آن را در ناحیه‌ای از فضا قرار می‌دهد که به‌طور همزمان تحت تأثیر دو جسم بزرگ است. این مسیرها نمی‌توانند از تحلیل دو جسم با اختلالات جسم سوم استفاده کنند، بلکه باید تأثیرات هر دو جسم به‌طور همزمان در نظر گرفته شوند.

مسئله سه‌جسمی محدود که شامل دو جسم اصلی با جرم‌های بزرگ و یک جسم کوچک (فضایپیما) است، محیطی مناسب برای بکارگیری روش‌های یادگیری تقویتی محسوب می‌شود. در این مسئله، دینامیک غیرخطی پیچیده‌ای حاکم است که نقاط تعادل خاصی به نام نقاط لاغرانژ در آن وجود دارد

در این فصل ابتدا به مدل‌سازی ریاضی مسئله سه‌جسمی محدود و استخراج معادلات حرکت در سیستم مختصات دوران در بخش ۱-۳ پرداخته شده است. سپس، نقاط لاغرانژ و خصوصیات پایداری آنها در بخش ۲-۳ مورد بررسی قرار گرفته.

#### ۱-۳ مسئله سه‌جسمی محدود دایره‌ای (CRTBP)

دو جرم اصلی (زمین با جرم  $m_1$  و ماه با جرم  $m_2$ ) روی مدارهای دایره‌ای و هم‌صفحه پیرامون مرکز جرم مشترک حرکت می‌کنند. جرم سوم (فضایپیما با جرم ناجیز  $m_3$ ) چنان کوچک فرض می‌شود که تأثیر گرانشی آن بر حرکت دو جسم اصلی قابل صرف نظر است؛ بدین ترتیب مسئله سه‌جسمی محدود دایره‌ای شکل می‌گیرد.

جدول ۱-۳: مقادیر عددی برای مسئله سه جسمی محدود (سیستم زمین-ماه)

مقدار عددی	توصیف	پارامتر
$5.972 \times 10^{24} \text{ kg}$	جرم زمین	$m_1$
$7.348 \times 10^{22} \text{ kg}$	جرم ماه	$m_2$
0.0121505856	نسبت جرمی	$\mu$
$2.6617 \times 10^{-6} \text{ rad/s}$	سرعت زاویه‌ای سیستم	$\omega$

دستگاه مختصات چرخانی همدوران با دو جرم اصلی انتخاب می‌شود؛ مبدأ در مرکز جرم سامانه است، محور  $x$  خطِ واصلِ دو جرم و محور  $y$  بر آن عمود (در صفحه‌ی مدارها) است. واحد طول برابر فاصله‌ی ثابت میان دو جرم و واحد زمان چنان تعریف می‌شود که دوره‌ی مداری سامانه  $2\pi$  (و در نتیجه  $1 = \omega$ ) گردد. همچنین جرم‌ها به‌گونه‌ای مقیاس می‌شود که مجموع دو جرم برابر با یک شود.

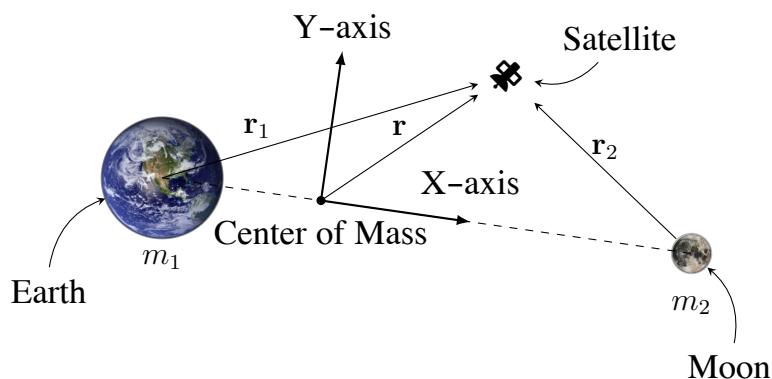
$$m_1 + m_2 = 1. \quad (1-3)$$

با نسبت جرمی

$$\mu \equiv \frac{m_2}{m_1 + m_2}, \quad (2-3)$$

داریم  $m_2 = \mu$  و  $m_1 = 1 - \mu$  و مکانِ دو جرم در دستگاه بی‌بعد به صورت

$$\mathbf{r}_{\text{Earth}} = (-\mu, 0), \quad \mathbf{r}_{\text{Moon}} = (1 - \mu, 0). \quad (3-3)$$



شکل ۱-۳: هندسه مسئله سه بدن محدود

### ۱-۱-۳ لاگرانژ و معادلات حرکت

با در نظر گرفتن  $G = 1$  در حالت بی بعد، تابع لاگرانژ جرم سوم در دستگاه چرخان برابر است با [۵۶]

$$L = \frac{1}{2}(\dot{x}^2 + \dot{y}^2 + \dot{z}^2) + (1 - \mu) \frac{1}{r_1} + \mu \frac{1}{r_2} + \frac{1}{2}(x^2 + y^2), \quad (4-3)$$

که در آن

$$r_1 = \sqrt{(x + \mu)^2 + y^2 + z^2}, \quad r_2 = \sqrt{(x - 1 + \mu)^2 + y^2 + z^2}. \quad (5-3)$$

با به کارگیری رابطه‌ی اویلر-لاگرانژ

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{q}_i} - \frac{\partial L}{\partial q_i} = 0, \quad q_i \in \{x, y, z\},$$

معادلات بی بعدِ حرکتِ جرم سوم به دست می‌آید:

$$\ddot{x} - 2\dot{y} = x - \frac{1 - \mu}{r_1^3}(x + \mu) - \frac{\mu}{r_2^3}(x - 1 + \mu), \quad (6-3)$$

$$\ddot{y} + 2\dot{x} = y - \frac{1 - \mu}{r_1^3}y - \frac{\mu}{r_2^3}y, \quad (7-3)$$

$$\ddot{z} = -\frac{1 - \mu}{r_1^3}z - \frac{\mu}{r_2^3}z. \quad (8-3)$$

یا به نگاشتِ برداری به صورت زیر است.

$$\ddot{\mathbf{r}} + 2\boldsymbol{\omega} \times \dot{\mathbf{r}} = \nabla \Omega(\mathbf{r}), \quad \Omega(x, y, z) = \frac{1}{2}(x^2 + y^2) + \frac{1 - \mu}{r_1} + \frac{\mu}{r_2}. \quad (9-3)$$

### ۲-۳ نقاط تعادل لاگرانژ

نقطه‌ی تعادل مکانی است که در چارچوب چرخان، جرم سوم بی‌حرکت بماند. این شرایط با صفرشدن مؤلفه‌های سرعت و شتاب به دست می‌آید؛ از این‌رو در معادلات بالا قرار می‌دهیم  $\dot{x} = \dot{y} = \dot{z} = \ddot{x} = \ddot{y} = \ddot{z} = 0$ . در

نتیجه دستگاه جبری زیر برای مختصات نقطه‌ی تعادل حاصل می‌شود:

$$0 = x - \frac{1 - \mu}{r_1^3}(x + \mu) - \frac{\mu}{r_2^3}(x - 1 + \mu), \quad (10-3)$$

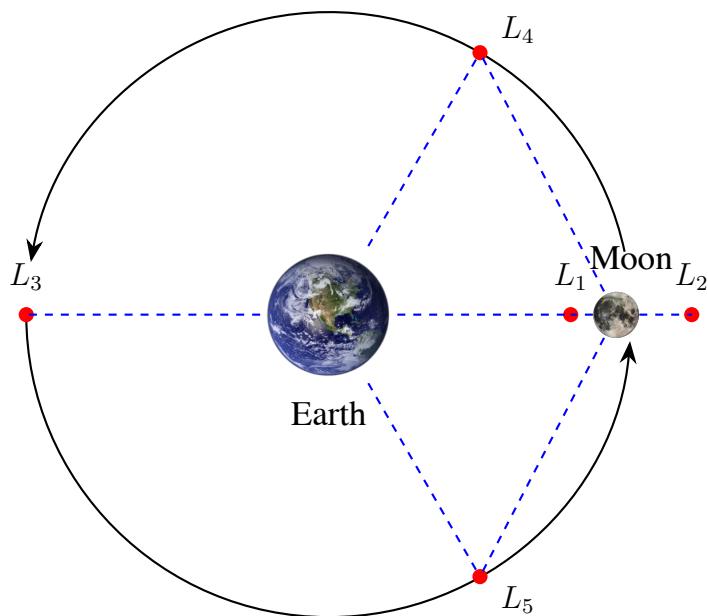
$$0 = y \left[ 1 - \frac{1 - \mu}{r_1^3} - \frac{\mu}{r_2^3} \right], \quad (11-3)$$

$$0 = -\frac{1 - \mu}{r_1^3}z - \frac{\mu}{r_2^3}z. \quad (12-3)$$

معادله‌ی سوم نشان می‌دهد برای حالت عمومی باید  $z = 0$  باشد؛ بنابراین نقاط تعادل همگی در صفحه‌ی مدار قرار می‌گیرند.

۱. نقاط هم خط (Collinear). سه نقطه‌ی  $L_1$ ,  $L_2$  و  $L_3$  روی خط واحد دو جرم قرار دارند و لذا  $y = 0$  است.

۲. نقاط سه‌گوش (Triangular). دو نقطه‌ی  $L_4$  و  $L_5$  رأس‌های مثلث متساوی‌الاضلاع با دو جرم اصلی را تشکیل می‌دهند و در آن‌ها  $y \neq 0$ .



شکل ۲-۳: نقاط لاغرانژ

### نقاط هم خط ( $L_1, L_2, L_3$ )

با اعمال  $y = 0$ , تنها معادله زیر باقی می‌ماند

$$x - \frac{1-\mu}{|x+\mu|^3}(x+\mu) - \frac{\mu}{|x-1+\mu|^3}(x-1+\mu) = 0. \quad (13-3)$$

این معادله در سه ناحیه‌ی مجزا—بین دو جرم، بیرون جرم کوچک و بیرون جرم بزرگ—دارای یک ریشه است که به ترتیب نقاط  $L_1$ ,  $L_2$  و  $L_3$  را تعیین می‌کند.

برای  $1 \ll \mu$  (همچون سامانه‌ی خورشید-زمین یا زمین-ماه) می‌توان تقریب‌های شناخته‌شده را نوشت:

$$x_{L_1} \simeq (1-\mu) - \left(\frac{\mu}{3}\right)^{1/3},$$

$$x_{L_2} \simeq (1-\mu) + \left(\frac{\mu}{3}\right)^{1/3},$$

$$x_{L_3} \simeq -1 - \frac{5}{12}\mu; \quad y_{L_i} = 0.$$

در عمل، ریشه‌ی دقیق معادله‌ی (۱۳-۳) با یک روش عددی (نیوتن-رافسون) محاسبه می‌شود.

### نقاط سه‌گوش ( $L_4, L_5$ )

در این نقاط  $1 = 1 - (1 - \mu)/r_1^3 - \mu/r_2^3 = 0$  و شرط  $r_1 = r_2 = 1$  به طور طبیعی برقرار است. مختصات به سادگی عبارت‌اند از

$$x_{L_4} = x_{L_5} = \frac{1}{2} - \mu, \quad y_{L_4} = +\frac{\sqrt{3}}{2}, \quad y_{L_5} = -\frac{\sqrt{3}}{2}. \quad (14-3)$$

پایداری این نقاط مستلزم نسبت جرم کافی است؛ شرط کلاسیک  $m_1/m_2 > 24.96$  در سامانه‌های خورشید-سیاره یا زمین-ماه به خوبی برقرار است و سبب وجود خانواده‌ی سیارک‌های تروجان حول  $L_4$  و  $L_5$  می‌شود. در مقابل، نقاط هم خط ناپایدارند و معمولاً مأموریت‌های فضایی روی مدارهای هاله‌ای یا لیساژور در پیرامون آن‌ها قرار می‌گیرند.

برای سامانه‌ی زمین-ماه،  $0.01215 \simeq \mu$  است. جدول زیر مختصات بی‌بعد هر پنج نقطه را نشان می‌دهد (واحد طول: فاصله‌ی زمین-ماه). موقعیت زمین در  $(0, 0, -\mu)$  و ماہ در  $(1 - \mu, 0, 0)$  است.

جدول ۲-۳: مقادیر عددی نقاط لاغرانژ برای مسئله سه‌جسمی محدود سیستم زمین-ماه

نقطه‌ی لاغرانژ	$x$ (بی‌بعد)	$y$ (بی‌بعد)
$L_1$	+0.83692	0
$L_2$	+1.15568	0
$L_3$	-1.00506	0
$L_4$	+0.48785	+0.86603
$L_5$	+0.48785	-0.86603

این نتایج نشان می‌دهد که  $L_1$  در حدود 0.84 فاصله‌ی زمین-ماه از زمین قرار دارد (فاصله‌ی آن تا ماہ در حدود 0.16 واحد طول است) و  $L_2$  بیرون مدار ماه است. نقطه‌ی  $L_3$  تقریباً یک واحد طول در سوی مقابله ماه نسبت به زمین قرار دارد. دو نقطه‌ی  $L_4$  و  $L_5$  در مختصات  $(0.488, \pm 0.866)$  قرار گرفته و با زمین و ماہ مثلث متساوی‌الاضلاع می‌سازند.

## فصل ۴

### یادگیری تقویتی

در این فصل به بررسی یادگیری تقویتی پرداخته شده است. ابتدا در فصل ۱-۴ مفاهیم اولیه یادگیری تقویتی ارائه شده است. در ادامه عامل‌های گرادیان سیاست عمیق قطعی ۲-۴، گرادیان سیاست عمیق قطعی تاخیری دوگانه ۳-۴، عملگر نقاد نرم ۴-۴ و بهینه‌سازی سیاست مجاور ۵-۴ توضیح داده شده است.

#### ۱-۴ مفاهیم اولیه

دو بخش اصلی یادگیری تقویتی<sup>۱</sup> شامل عامل<sup>۲</sup> و محیط<sup>۳</sup> است. عامل در محیط قرار دارد و با آن در تعامل است. در هر مرحله از تعامل بین عامل و محیط، عامل یک مشاهده جزئی از وضعیت محیط انجام می‌دهد و سپس در مورد اقدامی که باید انجام دهد، تصمیم می‌گیرد. وقتی عامل روی محیط عمل می‌کند، محیط تغییر می‌کند؛ اما، ممکن است محیط به تنها یک نیز تغییر کند. عامل همچنین یک سیگنال پاداش<sup>۴</sup> از محیط دریافت می‌کند؛ سیگنالی که به عامل می‌گوید وضعیت تعامل فعلی آن با محیط چقدر خوب یا بد است. هدف عامل بیشینه‌کردن پاداش انباسته خود است که برگشت<sup>۵</sup> نام دارد. یادگیری تقویتی به روش‌هایی گفته می‌شود که در آن‌ها عامل رفتارهای مناسب برای رسیدن به هدف خود را می‌آموزد. در شکل ۱-۴ تعامل بین محیط و عامل نشان داده شده است.

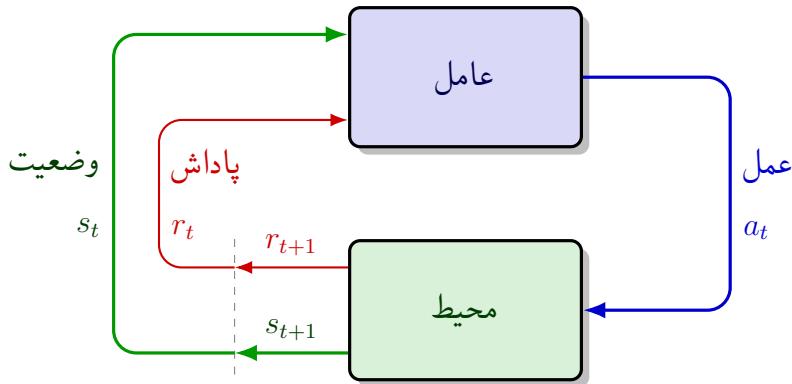
<sup>1</sup>Reinforcement Learning (RL)

<sup>2</sup>Agent

<sup>3</sup>Environment

<sup>4</sup>Reward

<sup>5</sup>Return



شکل ۱-۴: حلقه تعامل عامل و محیط

## ۱-۱-۴ حالت و مشاهدات

حالت<sup>۶</sup> ( $s$ ) توصیف کاملی از وضعیت محیط است. همه اطلاعات محیط در حالت وجود دارد. مشاهده<sup>۷</sup> ( $o$ ) یک توصیف جزئی از حالت است که ممکن است شامل تمامی اطلاعات نباشد. در این پژوهش مشاهده توصیف کاملی از محیط هست؛ در نتیجه، حالت و مشاهده برابر هستند.

## ۲-۱-۴ فضای عمل

فضای عمل ( $a$ ) در یادگیری تقویتی، مجموعه‌ای از تمام اقداماتی است که یک عامل می‌تواند در محیط انجام دهد. این فضا می‌تواند گسسته<sup>۸</sup> یا پیوسته<sup>۹</sup> باشد. در این پژوهش فضای عمل پیوسته و محدود به یک بازه مشخص است.

## ۳-۱-۴ سیاست

سیاست<sup>۱۰</sup> قاعده‌ای است که یک عامل برای تصمیم‌گیری در مورد اقدامات خود استفاده می‌کند. در این پژوهش به تناسب الگوریتم پیاده‌سازی شده از سیاست قطعی<sup>۱۱</sup> یا تصادفی<sup>۱۲</sup> استفاده شده است که به دو صورت زیر نشان

<sup>6</sup>State

<sup>7</sup>Observation

<sup>8</sup>Discrete

<sup>9</sup>Continuous

<sup>10</sup>Policy

<sup>11</sup>Deterministic

<sup>12</sup>Stochastic

داده می‌شود:

$$a_t = \mu(s_t) \quad (1-4)$$

$$a_t \sim \pi(\cdot | s_t) \quad (2-4)$$

که زیروند  $t$  بیانگر زمان است. در یادگیری تقویتی عمیق از سیاست‌های پارامتری شده استفاده می‌شود. خروجی این سیاست‌ها تابعی پارامترهای سیاست (وزن‌ها و بایاس‌های یک شبکه عصبی) هستند که می‌توان از الگوریتم‌های بهینه‌سازی جهت تعیین مقدار بهینه این پارامترها استفاده کرد. در این پژوهش پارامترهای سیاست با  $\theta$  نشان داده شده‌است و سپس نماد آن به عنوان زیروند سیاست مانند معادله (۳-۴) نشان داده شده‌است.

$$\begin{aligned} a_t &= \mu_\theta(s_t) \\ a_t &\sim \pi_\theta(\cdot | s_t) \end{aligned} \quad (3-4)$$

## ۴-۱-۴ مسیر

یک مسیر<sup>۱۳</sup> یک توالی از حالت‌ها و عمل‌ها در محیط است.

$$\tau = (s_0, a_0, s_1, a_1, \dots) \quad (4-4)$$

گذار حالت<sup>۱۴</sup> به اتفاقاتی که در محیط بین زمان  $t$  در حالت  $s_t$  و زمان  $t+1$  در حالت  $s_{t+1}$  رخ می‌دهد، گفته می‌شود. این گذارها توسط قوانین طبیعی محیط انجام می‌شوند و تنها به آخرین اقدام انجام‌شده توسط عامل  $(a_t)$  بستگی دارند. گذار حالت را می‌توان به صورت زیر تعریف کرد.

$$s_{t+1} = f(s_t, a_t) \quad (5-4)$$

## ۵-۱-۴ تابع پاداش و برگشت

تابع پاداش<sup>۱۵</sup> در حالت کلی به حالت فعلی محیط، آخرین عمل انجام‌شده و حالت بعدی محیط بستگی دارد. تابع پاداش را می‌توان به صورت زیر تعریف کرد.

$$r_t = R(s_t, a_t, s_{t+1}) \quad (6-4)$$

---

<sup>13</sup>Trajectory

<sup>14</sup>State Transition

<sup>15</sup>Reward Function

در این پژوهش، پاداش تنها تابعی از جفتِ حالت-عمل ( $r_t = R(s_t, a_t)$ ) فرض شده است. هدف عامل این است که مجموع پادash‌های به دست آمده در طول یک مسیر را به حداکثر برساند. در این پژوهش مجموع پادash‌ها در طول یک مسیر را با نماد  $R(\tau)$  نشان داده شده است و به آن تابع برگشت<sup>۱۶</sup> است که مجموع پادash‌های به دست آمده در یک بازه زمانی ثابت و از مسیر  $\tau$  است که در معادله (۷-۴) نشان داده شده است.

$$R(\tau) = \sum_{t=0}^T r_t \quad (7-4)$$

نوع دیگری از برگشت، برگشت تنزیل شده با افق نامحدود<sup>۱۹</sup> است که مجموع همه پادash‌هایی است که تا به حال توسط عامل به دست آمده است. اما، فاصله زمانی تا دریافت پاداش باعث تنزیل ارزش آن می‌شود. این معادله برگشت (۸-۴) شامل یک فاکتور تنزیل<sup>۲۰</sup> با نماد  $\gamma$  است که عددی بین صفر و یک است.

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \quad (8-4)$$

## ۶-۱-۶ ارزش در یادگیری تقویتی

در یادگیری تقویتی، دانستن ارزش<sup>۲۱</sup> یک حالت یا جفتِ حالت-عمل ضروری است. منظور از ارزش، برگشت مورد انتظار<sup>۲۲</sup> است. یعنی اگر از آن حالت یا جفت حالت-عمل شروع شود و سپس برای همیشه طبق یک سیاست خاص عمل شود، به طور میانگین چه مقدار پاداش دریافت خواهد شد. توابع ارزش تقریباً در تمام الگوریتم‌های یادگیری تقویتی به کار می‌روند. در اینجا به چهار تابع مهم اشاره شده است.

۱. تابع ارزش تحت سیاست<sup>۲۳</sup> ( $V^\pi(s)$ ): خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت  $s$  شروع شود و همیشه طبق سیاست  $\pi$  عمل شود و به صورت زیر بیان می‌شود:

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s] \quad (9-4)$$

۲. تابع ارزش-عمل تحت سیاست<sup>۲۴</sup> ( $Q^\pi(s, a)$ ): خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت  $s$  شروع شود، یک اقدام دلخواه  $a$  (که ممکن است از سیاست  $\pi$  نباشد) انجام شود و سپس

<sup>16</sup>Return

<sup>17</sup>Discount

<sup>18</sup>Finite-Horizon Undiscounted Return

<sup>19</sup>Infinite-Horizon Discounted Return

<sup>20</sup>Discount Factor

<sup>21</sup>Value

<sup>22</sup>Expected Return

<sup>23</sup>On-Policy Value Function

<sup>24</sup>On-Policy Action-Value Function

برای همیشه طبق سیاست  $\pi$  عمل شود و بهصورت زیر بیان می‌شود:

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s, a_0 = a] \quad (10-4)$$

۳. تابع ارزش بهینه<sup>۲۵</sup> ( $V^*(s)$ ): خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت  $s$  شروع شود و همیشه طبق سیاست بهینه در محیط عمل شود و بهصورت زیر بیان می‌شود:

$$V^*(s) = \max_{\pi} (V^\pi(s)) \quad (11-4)$$

۴. تابع ارزش-عمل بهینه<sup>۲۶</sup> ( $Q^*(s, a)$ ): خروجی این تابع برگشت مورد انتظار است در صورتی که از حالت  $s$  شروع شود، یک اقدام دلخواه  $a$  انجام شود و سپس برای همیشه طبق سیاست بهینه در محیط عمل شود و بهصورت زیر بیان می‌شود:

$$Q^*(s, a) = \max_{\pi} (Q^\pi(s, a)) \quad (12-4)$$

## ۷-۱-۴ معادلات بلمن

توابع ارزش اشاره شده از معادلات خاصی که به آنها معادلات بلمن گفته می‌شود، پیروی می‌کنند. ایده اصلی پشت معادلات بلمن این است که ارزش نقطه شروع برابر است با پاداشی است که انتظار دارید از آنجا دریافت کنید، به علاوه ارزش مکانی که بعداً به آنجا می‌رسید. معادلات بلمن برای توابع ارزش سیاست محور به شرح زیر هستند:

$$V^\pi(s) = \mathbb{E}_{\substack{a \sim \pi \\ s' \sim P}} [r(s, a) + \gamma V^\pi(s')] \quad (13-4)$$

$$Q^\pi(s, a) = r(s, a) + \mathbb{E}_{\substack{a \sim \pi \\ s' \sim P}} \left[ \gamma \mathbb{E}_{a' \sim \pi} [Q^\pi(s', a')] \right] \quad (14-4)$$

که در آن  $V^\pi(s)$  تابع ارزش حالت  $s$  تحت سیاست  $\pi$  است؛  $Q^\pi(s, a)$  تابع ارزش عمل  $a$  در حالت  $s$  تحت سیاست  $\pi$  است؛  $R(s, a)$  پاداش دریافتی پس از انجام عمل  $a$  در حالت  $s$  است؛  $\gamma$  ضریب تنزیل است که ارزش پاداش‌های آینده را کاهش می‌دهد؛  $P(\cdot | s, a)$  نشان می‌دهد که حالت بعدی  $s'$  از توزیع انتقال محیط  $P$  با شرطهای  $s$  و  $a$  نمونه‌برداری می‌شود؛ و  $\pi(\cdot | s')$  نشان می‌دهد که عمل بعدی  $a'$  از سیاست

<sup>25</sup>Optimal Value Function

<sup>26</sup>Optimal Action-Value Function

$\pi$  با شرط حالت جدید  $s'$  نمونه برداری می شود. این معادلات بیانگر این هستند که ارزش یک حالت یا عمل، مجموع پاداش مورد انتظار آن و ارزش حالت بعدی است که بر اساس سیاست فعلی تعیین می شود. معادلات بلمن برای توابع ارزش بهینه به شرح زیر هستند:

$$V^*(s) = \max_a \mathbb{E}_{s' \sim P} [r(s, a) + \gamma V^*(s')] \quad (15-4)$$

$$Q^*(s, a) = r(s, a) + \mathbb{E}_{s' \sim P} \left[ \gamma \max_{a'} Q^*(s', a') \right] \quad (16-4)$$

تفاوت حیاتی بین معادلات بلمن برای توابع ارزش سیاست محور و توابع ارزش بهینه، عدم حضور یا حضور  $\max$  بر روی اعمال است. حضور آن منعکس کننده این است که هرگاه عامل بتواند عمل خود را انتخاب کند، برای عمل بهینه، باید هر عملی را که منجر به بالاترین ارزش می شود انتخاب کند.

## ۸-۱-۴ تابع مزیت

گاهی در یادگیری تقویتی، نیازی به توصیف میزان خوبی یک عمل به صورت مطلق نیست، بلکه تنها می خواهیم بدانیم که چه مقدار بهتر از سایر اعمال به طور متوسط است. به عبارت دیگر، مزیت نسبی آن عمل مورد بررسی قرار می گیرد. این مفهوم با تابع مزیت <sup>۲۷</sup> توضیح داده می شود.

تابع مزیت  $A^\pi(s, a)$  که مربوط به سیاست  $\pi$  است، توصیف می کند که انجام یک عمل خاص  $a$  در حالت  $s$  چقدر بهتر از انتخاب تصادفی یک عمل بر اساس  $(\cdot | s)^\pi$  است، با فرض اینکه شما برای همیشه پس از آن مطابق با  $\pi$  عمل می کنید. به صورت ریاضی، تابع مزیت به صورت زیر تعریف می شود:

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

که در آن  $A^\pi(s, a)$  تابع مزیت برای عمل  $a$  در حالت  $s$  است.  $Q^\pi(s, a)$  تابع ارزش عمل  $a$  در حالت  $s$  تحت سیاست  $\pi$  است و  $V^\pi(s)$  تابع ارزش حالت  $s$  تحت سیاست  $\pi$  است. این تابع مزیت نشان می دهد که انجام عمل  $a$  در حالت  $s$  نسبت به میانگین اعمال تحت سیاست  $\pi$  چقدر مزیت دارد. اگر  $A^\pi(s, a)$  مثبت باشد، نشان دهنده این است که عمل  $a$  بهتر از میانگین اعمال است و اگر منفی باشد، نشان دهنده کمتر بودن عمل کرد آن نسبت به میانگین است.

---

<sup>27</sup> Advantage Function

## ۲-۴ عامل گرادیان سیاست عمیق قطعی

گرادیان سیاست عمیق قطعی<sup>۲۸</sup> الگوریتمی است که هم‌مان یک تابع  $Q$  و یک سیاست را یاد می‌گیرد. این الگوریتم برای یادگیری تابع  $Q$  از داده‌های غیرسیاست محور<sup>۲۹</sup> و معادله بلمن استفاده می‌کند. این الگوریتم برای یادگیری سیاست نیز از تابع  $Q$  استفاده می‌کند.

این رویکرد وابستگی نزدیکی به یادگیری  $Q$  دارد. اگر تابع ارزش-عمل بهینه مشخص باشد، در هر حالت داده‌شده عمل بهینه را می‌توان با حل معادله (۱۷-۴) به دست آورد.

$$a^*(s) = \arg \max_a Q^*(s, a) \quad (17-4)$$

الگوریتم DDPG ترکیبی از یادگیری تقریبی برای  $(s, a)^*$  و یادگیری تقریبی برای  $(s)$  است و به صورتی طراحی شده است که برای محیط‌هایی با فضاهای عمل پیوسته مناسب باشد. آنچه این الگوریتم را برای فضای عمل پیوسته مناسب می‌کند، روش محاسبه  $(s, a)^*$  است. فرض می‌شود که تابع  $(s)$  نسبت به آرگومان عمل مشتق‌پذیر است. مشتق‌پذیری این امکان را می‌دهد که یک روش یادگیری مبتنی بر گرادیان برای سیاست  $(s)$  استفاده شود. سپس، به جای اجرای یک بهینه‌سازی زمانبر در هر بار محاسبه  $\max_a Q(s, a)$ ، می‌توان آن را با رابطه  $\max_a Q(s, a) \approx Q(s, \mu(s))$  تقریب زد.

## ۱-۴ یادگیری $Q$ در DDPG

معادله بلمن که تابع ارزش عمل بهینه  $(Q^*(s, a))$  را توصیف می‌کند، در پایین آورده شده است.

$$Q^*(s, a) = r(s, a) + \mathbb{E}_{s' \sim P} \left[ \gamma \max_{a'} Q^*(s', a') \right] \quad (18-4)$$

عبارت  $P \sim s'$  به این معنی است که وضعیت بعدی یعنی  $s'$  از توزیع احتمال  $P(\cdot | s, a)$  نمونه گرفته می‌شود. در معادله بلمن نقطه شروع برای یادگیری  $(s, a)^*$  یک مقداردهی تقریبی است. پارامترهای شبکه عصبی  $Q_\phi(s, a)$  با علامت  $\phi$  نشان داده شده است. مجموعه  $\mathcal{D}$  شامل اطلاعات جمع‌آوری شده تغییر از یک حالت به حالت دیگر  $(s, a, r, s', d)$  (که  $d$  نشان می‌دهد که آیا وضعیت  $s'$  پایانی است یا خیر) است. در بهینه‌سازی از تابع خطای میانگین مربعات بلمن<sup>۳۰</sup> (MSBE) استفاده شده است که معیاری برای نزدیکی  $Q_\phi$  به حالت بهینه برای برآورده کردن معادله بلمن است.

<sup>28</sup>Deep Deterministic Policy Gradient (DDPG)

<sup>29</sup>Off-Policy

<sup>30</sup>Mean Squared Bellman Error

$$L(\phi, \mathcal{D}) = \underset{(s, a, r, s', d) \sim \mathcal{D}}{\mathbb{E}} \left[ \left( Q_\phi(s, a) - \left( r + \gamma(1-d) \max_{a'} Q_\phi(s', a') \right) \right)^2 \right] \quad (19-4)$$

در الگوریتم DDPG دو ترفندهای عمکرد بهتر استفاده شده است که در ادامه به بررسی آن پرداخته شده است.

#### • بافرهای تکرار بازی

الگوریتم‌های یادگیری تقویتی جهت آموزش یک شبکه عصبی عمیق برای تقریب  $Q^*(s, a)$  از بافرهای تکرار بازی<sup>31</sup> تجربه شده استفاده می‌کنند. این مجموعه  $\mathcal{D}$  شامل تجربیات قبلی عامل است. برای داشتن رفتار پایدار در الگوریتم، بافر تکرار بازی باید به اندازه کافی بزرگ باشد تا شامل یک دامنه گسترده از تجربیات شود. انتخاب داده‌های بافر به دقت انجام شده است چرا که اگر فقط از داده‌های بسیار جدید استفاده شود، بیش برآراش<sup>32</sup> رخ می‌دهید و اگر از تجربه بیش از حد استفاده شود، ممکن است فرآیند یادگیری کند شود.

#### • شبکه‌های هدف

الگوریتم‌های یادگیری  $Q$  از شبکه‌های هدف استفاده می‌کنند. اصطلاح زیر به عنوان هدف شناخته می‌شود.

$$r + \gamma(1-d) \max_{a'} Q_\phi(s', a') \quad (20-4)$$

در هنگام کمینه کردن تابع خطای میانگین مربعات بلمن، سعی شده است تا تابع  $Q$  شبیه‌تر به هدف یعنی رابطه<sup>20-4</sup> شود. اما مشکل این است که هدف بستگی به پارامترهای در حال آموزش  $\phi$  دارد. این باعث ایجاد ناپایداری در کمینه کردن تابع خطای میانگین مربعات بلمن می‌شود. راه حل آن استفاده از یک مجموعه پارامترهایی است که با تأخیر زمانی به  $\phi$  نزدیک می‌شوند. به عبارت دیگر، یک شبکه دوم ایجاد می‌شود که به آن شبکه هدف گفته می‌شود. شبکه هدف پارامترهای شبکه اول را با تاخیر دنبال می‌کند. پارامترهای شبکه هدف با نشان targ $\phi$  نشان داده می‌شوند. در الگوریتم DDPG، شبکه هدف در هر بهروزرسانی شبکه اصلی، با میانگین‌گیری پولیاک<sup>33</sup> به صورت زیر بهروزرسانی می‌شود.

$$\phi_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1-\rho) \phi \quad (21-4)$$

در رابطه بالا  $\rho$  یک ابرپارامتر<sup>34</sup> است که بین صفر و یک انتخاب می‌شود. در این پژوهش این مقدار نزدیک به یک درنظر گرفته شده است.

<sup>31</sup>Replay Buffers

<sup>32</sup>Overfit

<sup>33</sup>Polyak Averaging

<sup>34</sup>Hyperparameter

الگوریتم DDPG نیاز به یک شبکه سیاست هدف ( $\mu_{\theta_{\text{targ}}}$ ) برای محاسبه عملهایی که به طور تقریبی بیشینه  $Q$  را حاصل کند، را دارد. برای رسیدن به این شبکه سیاست هدف از همان روشی که تابع  $Q$  به دست می‌آید یعنی با میانگین‌گیری پولیاک از پارامترهای سیاست در طول زمان آموزش استفاده می‌شود.

با درنظرگرفتن موارد اشاره شده، یادگیری  $Q$  در DDPG با کمینه کردن تابع خطای میانگین مربعات بلمن (معادله ۲۲-۴) با استفاده از کاهش گرادیان تصادفی<sup>۳۵</sup> (MSBE) می‌شود.

$$L(\phi, \mathcal{D}) = \underset{(s, a, r, s', d) \sim \mathcal{D}}{\mathbb{E}} \left[ \left( Q_\phi(s, a) - (r + \gamma(1-d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s')) \right)^2 \right] \quad (22-4)$$

## ۲-۲-۴ سیاست در DDPG

در این بخش یک سیاست تعیین شده ( $\mu_\theta(s)$ ) یادگرفته می‌شود تا عملی را انجام می‌دهد که بیشینه  $Q_\phi(s, a)$  رخ دهد. از آنجا که فضای عمل پیوسته است و فرض شده است که تابع  $Q$  نسبت به عمل مشتق‌پذیر است، رابطه زیر با استفاده از صعود گرادیان<sup>۳۶</sup> ( تنها نسبت به پارامترهای سیاست) بیشینه می‌شود.

$$\max_{\theta} \underset{s \sim \mathcal{D}}{\mathbb{E}} [Q_\phi(s, \mu_\theta(s))] \quad (23-4)$$

## ۳-۲-۴ اکتشاف و بهره‌برداری در DDPG

برای بهبود اکتشاف<sup>۳۷</sup> در سیاست‌های DDPG، در زمان آموزش نویز به عمل‌ها اضافه می‌شود. نویسنده‌گان مقاله DDPG<sup>۳۸</sup> [۵۷] توصیه کرده‌اند که نویز OU<sup>۳۹</sup> با همبندی زمانی<sup>۴۰</sup> اضافه شود. در زمان بهره‌برداری سیاست، از آنچه یادگرفته است، نویز به عمل‌ها اضافه نمی‌شود.

## ۴-۲-۴ شبکه DDPG

در این بخش، شبکه DDPG پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم ۱ در محیط پایتون با استفاده از کتابخانه TensorFlow<sup>۴۱</sup> [۵۸] پیاده‌سازی شده است.

<sup>35</sup>Stochastic Gradient Descent

<sup>36</sup>Gradient Ascent

<sup>37</sup>Exploration

<sup>38</sup>Ornstein–Uhlenbeck

<sup>39</sup>Time-Correlated

<sup>40</sup>Exploitation

---

## الگوریتم ۱ گرادیان سیاست عمیق قطعی

---

- وروودی: پارامترهای اولیه سیاست  $(\theta)$ ، پارامترهای تابع  $Q(\phi)$ ، بافر تکرار بازی خالی  $(\mathcal{D})$
- ۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید  $\theta \leftarrow \theta_{\text{targ}}$
  - ۲: تا وقتی همگرایی رخ دهد:
  - ۳: وضعیت  $s$  را مشاهده کرده و عمل  $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{\text{Low}}, a_{\text{High}})$  را انتخاب کنید به طوری که  $\mathcal{N} \sim \epsilon$  است.
  - ۴: عمل  $a$  را در محیط اجرا کنید.
  - ۵: وضعیت بعدی  $s'$ ، پاداش  $r$  و سیگنال پایان  $d$  را مشاهده کنید تا نشان دهد آیا  $s'$  پایانی است یا خیر.
  - ۶: اگر  $s'$  پایانی است، وضعیت محیط را بازنشانی کنید.
  - ۷: اگر زمان بهروزرسانی فرا رسیده است:
  - ۸: به ازای هر تعداد بهروزرسانی:
  - ۹: یک دسته تصادفی گذر از یک حالت به حالت دیگر، از  $\mathcal{D}$ ، از  $B = \{(s, a, r, s', d)\}$ ، از  $\mathcal{N}$  نمونه‌گیری شود.
  - ۱۰: هدف را محاسبه کنید:

$$y(r, s', d) = r + \gamma(1 - d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))$$

تابع  $Q$  را با یک مرحله از نزول گرادیان با استفاده از رابطه زیر بهروزرسانی کنید: ۱۱

$$\nabla_\phi \frac{1}{|B|} \sum_{(s, a, r, s', d) \in B} (Q_\phi(s, a) - y(r, s', d))^2$$

سیاست را با یک مرحله از صعود گرادیان با استفاده از رابطه زیر بهروزرسانی کنید: ۱۲

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} Q_\phi(s, \mu_\theta(s))$$

شبکه‌های هدف را با استفاده از معادلات زیر بهروزرسانی کنید: ۱۳

$$\phi_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1 - \rho) \phi$$

$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho) \theta$$

## ۳-۴ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه

عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه<sup>۴۱</sup> یکی از الگوریتم‌های یادگیری تقویتی است که برای حل مسائل کنترل در محیط‌های پیوسته طراحی شده است. این الگوریتم بر اساس الگوریتم DDPG توسعه یافته و با استفاده از تکنیک‌های مختلف، پایداری و کارایی یادگیری را بهبود می‌بخشد. در حالی که DDPG گاهی اوقات می‌تواند عملکرد بسیار خوبی داشته باشد، اما اغلب نسبت به ابرپارامترها و سایر انواع تنظیمات یادگیری حساس است. یک حالت رایج شکستِ عامل DDPG در یادگیری این است که تابع  $Q$  یادگرفته شده شروع به بیش برآورد مقادیر  $Q$  می‌کند که منجر به واگرایی سیاست می‌شود. واگرایی به این دلیل رخ می‌دهد که در فرآیند یادگیری سیاست از تخمین تابع  $Q$  استفاده می‌شود که افزایش خطای تابع  $Q$  منجر به ناپایداری در یادگیری سیاست می‌شود.

الگوریتم (Twin Delayed DDPG) از دو ترفندهای زیر جهت بهبود مشکلات اشاره شده استفاده می‌کند.

- یادگیری دوگانه محدودشده<sup>۴۲</sup>: الگوریتم TD3 به جای یک تابع  $Q$ ، دو تابع  $Q_{\phi_1}$  و  $Q_{\phi_2}$  را یاد می‌گیرد (از این رو دوگانه<sup>۴۳</sup> نامیده می‌شود) و از کوچکترین مقدار این دو  $Q_{\phi_1}$  و  $Q_{\phi_2}$  در تابع بلمن استفاده می‌شود. نحوه محاسبه هدف بر اساس دو تابع  $Q$  اشاره شده در رابطه (۲۴-۴) آورده شده است.

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_i, \text{targ}}(s', a'(s')) \quad (24-4)$$

سپس، در هر دو تابع  $Q_{\phi_1}$  و  $Q_{\phi_2}$  یادگیری انجام می‌شود.

$$L(\phi_1, \mathcal{D}) = \mathbb{E}_{(s, a, r, s', d) \sim \mathcal{D}} \left( Q_{\phi_1}(s, a) - y(r, s', d) \right)^2 \quad (25-4)$$

$$L(\phi_2, \mathcal{D}) = \mathbb{E}_{(s, a, r, s', d) \sim \mathcal{D}} \left( Q_{\phi_2}(s, a) - y(r, s', d) \right)^2 \quad (26-4)$$

- بهروزرسانی‌های تاخیری سیاست<sup>۴۴</sup>: الگوریتم TD3 سیاست را با تاخیر بیشتری نسبت به تابع  $Q$  بهروزرسانی می‌کند. در مرجع [۵۹] توصیه شده است که برای هر دو بهروزرسانی تابع  $Q$ ، یک بهروزرسانی سیاست انجام شود.

<sup>41</sup>Twin Delayed Deep Deterministic Policy Gradient (TD3)

<sup>42</sup>Clipped Double-Q Learning

<sup>43</sup>twin

<sup>44</sup>Delayed Policy Updates

این دو ترفند منجر به بهبود قابل توجه عملکرد TD3 نسبت به DDPG پایه می‌شوند. در نهایت سیاست با بیشینه‌کردن  $Q_{\phi_1}$  آموخته می‌شود:

$$\max_{\theta} \mathbb{E}_{s \sim \mathcal{D}} [Q_{\phi_1}(s, \mu_{\theta}(s))] \quad (27-4)$$

### ۱-۳-۴ اکتشاف و بهره‌برداری در TD3

الگوریتم TD3 یک سیاست قطعی را به صورت غیرسیاست محور آموزش می‌دهد. از آنجایی که سیاست قطعی است، در ابتدا عامل تنوع کافی از اعمال را برای یافتن روش‌های مفید امتحان نمی‌کند. برای بهبود اکتشاف سیاست‌های TD3، در زمان آموزش نویز به عمل‌ها اضافه می‌شود. در این پژوهش، نویز گاوی با میانگین صفر بدون همبندی زمانی اعمال شده است. شدت نویز جهت بهره‌برداری بهتر در طول زمان کاهش می‌یابد.

### ۲-۳-۴ TD3 شبکه

در این بخش الگوریتم TD3 پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم [۴](#) در محیط پایتون با استفاده از کتابخانه PyTorch [\[۶۰\]](#) پیاده‌سازی شده است.

---

## الگوریتم ۲ عامل گرادیان سیاست عمیق قطعی تا خیری دوگانه

---

- وروودی: پارامترهای اولیه سیاست  $(\theta)$ ، پارامترهای تابع  $Q(\phi_1, \phi_2)$ ، بافر بازی خالی  $(\mathcal{D})$
- ۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید  $\theta_{\text{targ}} \leftarrow \theta$ ,  $\phi_{\text{targ},2} \leftarrow \phi_2$ ,  $\phi_{\text{targ},1} \leftarrow \phi_1$
  - ۲: تا وقتی همگرایی رخ دهد:
  - ۳: وضعیت  $(s)$  را مشاهده کرده و عمل  $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{\text{Low}}, a_{\text{High}})$  را انتخاب کنید، به طوری که  $\mathcal{N} \sim \epsilon$  است.
  - ۴: عمل  $a$  را در محیط اجرا کنید.
  - ۵: وضعیت بعدی  $s'$ ، پاداش  $r$  و سیگنال پایان  $d$  را مشاهده کنید تا نشان دهد آیا  $s'$  پایانی است یا خیر.
  - ۶: اگر  $s'$  پایانی است، وضعیت محیط را بازنشانی کنید.
  - ۷: اگر زمان بهروزرسانی فرا رسیده است:
  - ۸: به ازای  $j$  در هر تعداد بهروزرسانی:
  - ۹: یک دسته تصادفی گذر از یک حالت به حالت دیگر، از  $\mathcal{D}$ ،  $B = \{(s, a, r, s', d)\}$ ، از
  - ۱۰: هدف را محاسبه کنید:

$$y(r, s', d) = r + \gamma(1 - d) \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', a'(s'))$$

تابع  $Q$  را با یک مرحله از نزول گرادیان با استفاده از رابطه زیر بهروزرسانی کنید: ۱۱

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s, a) - y(r, s', d))^2 \quad \text{for } i = 1, 2$$

اگر باقیمانده  $j$  بر تا خیر سیاست برابر ۰ باشد: ۱۲

سیاست را با یک مرحله از صعود گرادیان با استفاده از رابطه زیر بهروزرسانی کنید: ۱۳

$$\nabla_{\theta} \frac{1}{|B|} \sum_{s \in B} Q_{\phi_1}(s, \mu_{\theta}(s))$$

شبکه‌های هدف را با استفاده از معادلات زیر بهروزرسانی کنید: ۱۴

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho) \phi_i \quad \text{for } i = 1, 2$$

$$\theta_{\text{targ}} \leftarrow \rho \theta_{\text{targ}} + (1 - \rho) \theta$$

## ۴-۴ عامل عملگر نقاد نرم

عملگرد نقاد نرم<sup>۴۵</sup> الگوریتمی است که یک سیاست تصادفی را به صورت غیرسیاست محور بهینه می‌کند و پلی بین بهینه‌سازی سیاست تصادفی و رویکردهای غیرسیاست محور مانند DDPG ایجاد می‌کند. این الگوریتم جانشین مستقیم TD3 نیست (زیرا تقریباً همزمان منتشر شده است)؛ اما، ترفند یادگیری دوگانه محدود شده را در خود جای داده است و به دلیل سیاست تصادفی SAC، از روشی به نام صاف کردن سیاست هدف<sup>۴۶</sup> استفاده شده است. یکی از ویژگی‌های اصلی SAC، تنظیم آنتروپی است. آنتروپی معیاری از تصادفی بودن انتخاب عمل در سیاست است. آموزش سیاست در جهت تعادل بهینه بین آنتروپی و بیشنه‌سازی بازده مورد انتظار است. این شرایط ارتباط نزدیکی با تعادل اکتشاف-بهره‌برداری دارد. افزایش آنتروپی منجر به اکتشاف بیشتر می‌شود که می‌تواند یادگیری را در مراحل بعدی تسريع کند. همچنین، می‌تواند از همگرایی زودهنگام سیاست به یک بهینه محلی بد جلوگیری کند. برای توضیح SAC، ابتدا باید به بررسی یادگیری تقویتی تنظیم شده با آنتروپی<sup>۴۷</sup> پرداخته شود. در RL تنظیم شده با آنتروپی، روابط تابع ارزش کمی متفاوت است.

### ۱-۴-۴ یادگیری تقویتی تنظیم شده با آنتروپی

آنتروپی معیاری برای سنجش میزان عدم قطعیت یا تصادفی بودن یک متغیر تصادفی یا توزیع احتمال آن است. به عبارت دقیق‌تر، آنتروپی برای یک توزیع احتمال، میانگین اطلاعات حاصل از نمونه‌برداری از آن توزیع را اندازه‌گیری می‌کند. در زمینه یادگیری تقویتی، تنظیم با آنتروپی تکنیکی است که با افزودن یک ترم متناسب با آنتروپی سیاست به تابع هدف، عامل را تشویق به اکتشاف بیشتر و اتخاذ سیاست‌های تصادفی‌تر می‌کند. این امر می‌تواند به بهبود پایداری فرآیند یادگیری و جلوگیری از همگرایی زودهنگام به بهینه‌های محلی کمک کند. فرض کنید  $X$  یک متغیر تصادفی پیوسته با تابع چگالی احتمال  $p(x)$  باشد. آنتروپی  $H(X)$  این متغیر تصادفی به صورت امید ریاضی لگاریتم منفی چگالی احتمال آن تعریف می‌شود:

$$H(X) = \text{E}_{x \sim p} [-\log p(x)] \quad (28-4)$$

### ۲-۴-۴ سیاست در SAC

در یادگیری تقویتی تنظیم شده با آنتروپی، عامل در هر مرحله زمانی متناسب با آنتروپی سیاست در آن مرحله زمانی پاداش دریافت می‌کند. بر اساس توضیحات اشاره شده روابط یادگیری تقویتی به صورت زیر می‌شود.

<sup>45</sup>Soft Actor Critic (SAC)

<sup>46</sup>Target Policy Smoothing

<sup>47</sup>Entropy-Regularized Reinforcement Learning

$$\pi^* = \arg \max_{\pi} \mathbb{E}_{\tau \sim \pi} \sum_{t=0}^{\infty} \gamma^t \left( R(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot | s_t)) \right) \quad (29-4)$$

که در آن ( $\alpha > 0$ ) ضریب مبادله<sup>48</sup> است.

### ۳-۴-۴ تابع ارزش در SAC

اکنون می‌توان تابع ارزش کمی متفاوت را بر اساس این مفهوم تعریف کرد.  $V^\pi$  به گونه‌ای تغییر می‌کند که پاداش‌های آنتروپی را از هر مرحله زمانی شامل می‌شود.

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t \left( R(s_t, a_t, s_{t+1}) + \alpha H(\pi(\cdot | s_t)) \right) \middle| s_0 = s \right] \quad (30-4)$$

### ۴-۴-۴ تابع Q در SAC

تابع  $Q^\pi$  به گونه‌ای تغییر می‌کند که پاداش‌های آنتروپی را از هر مرحله زمانی به جز مرحله اول شامل می‌شود.

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) + \alpha \sum_{t=1}^{\infty} \gamma^t H(\pi(\cdot | s_t)) \middle| s_0 = s, a_0 = a \right] \quad (31-4)$$

با این تعاریف رابطه<sup>4</sup>  $V^\pi$  و  $Q^\pi$  به صورت زیر است.

$$V^\pi(s) = \mathbb{E}_{a \sim \pi} [Q^\pi(s, a)] + \alpha H(\pi(\cdot | s)) \quad (32-4)$$

### ۵-۴-۴ معادله بلمن در SAC

معادله بلمن در حالت تنظیم شده با آنتروپی به صورت زیر ارائه می‌شود.

$$Q^\pi(s, a) = \mathbb{E}_{\substack{s' \sim P \\ a' \sim \pi}} [R(s, a, s') + \gamma (Q^\pi(s', a') + \alpha H(\pi(\cdot | s')))] \quad (33-4)$$

$$= \mathbb{E}_{s' \sim P} [R(s, a, s') + \gamma V^\pi(s')] \quad (34-4)$$

---

<sup>48</sup>Trade-Off

## ۶-۴-۴ یادگیری Q

با درنظرگرفتن موارد اشاره شده، یادگیری Q در SAC با کمینه کردن تابع خطای میانگین مربعات بلمن (MSBE) یعنی معادله (۳۵-۴) با استفاده از کاهش گرادیان انجام می شود.

$$L(\phi_i, \mathcal{D}) = \underset{(s,a,r,s',d) \sim \mathcal{D}}{\text{E}} \left[ \left( Q_{\phi_i}(s, a) - y(r, s', d) \right)^2 \right] \quad (35-4)$$

در معادله (۳۵-۴) تابع هدف برای روش یادگیری تقویتی SAC به صورت زیر تعریف می شود.

$$y(r, s', d) = r + \gamma(1-d) \left( \min_{j=1,2} Q_{\phi_{\text{targ},j}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s') \right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s') \quad (36-4)$$

نماد عمل بعدی را به جای  $a'$  به  $\tilde{a}'$  تغییر داده شده تا مشخص شود که عمل های بعدی باید از آخرین سیاست نمونه برداری شوند در حالی که  $r$  و  $s$  باید از بافر تکرار بازی آمده باشند.

## ۷-۴-۴ سیاست در SAC

سیاست باید در هر وضعیت برای به حداکثر رساندن بازگشت مورد انتظار آینده به همراه آنتروپی مورد انتظار آینده عمل کند. یعنی باید  $V^\pi(s)$  را به حداکثر برساند، بسط تابع ارزش در ادامه آمده است.

$$V^\pi(s) = \underset{a \sim \pi}{\text{E}} [Q^\pi(s, a)] + \alpha H(\pi(\cdot|s)) \quad (37-4)$$

$$= \underset{a \sim \pi}{\text{E}} [Q^\pi(s, a) - \alpha \log \pi(a|s)] \quad (38-4)$$

در بهینه سازی سیاست از ترفندهای پارامترسازی مجدد<sup>۴۹</sup> استفاده می شود، که در آن نمونه ای از  $\pi_\theta(\cdot|s)$  با محاسبه یک تابع قطعی از وضعیت، پارامترهای سیاست و نویز مستقل استخراج می شود. در این پژوهش مانند نویسندهای مقاله SAC [۶۱]، از یک سیاست گاووسی فشرده<sup>۵۰</sup> استفاده شده است. بر اساس این روش نمونه ها مطابق با رابطه زیر بدست می آیند:

$$\tilde{a}_\theta(s, \xi) = \tanh(\mu_\theta(s) + \sigma_\theta(s) \odot \xi), \quad \xi \sim \mathcal{N} \quad (39-4)$$

در رابطه بالا  $\odot$  نماد ضرب داخلی است. تابع  $\tanh$  در سیاست SAC تضمین می کند که اعمال در یک محدوده متناهی محدود شوند. این مورد در سیاست های VPG، TRPO و PPO وجود ندارد. همچنین اعمال این تابع توزیع را از حالت گاووسی تغییر می دهد.

<sup>49</sup>Reparameterization

<sup>50</sup>Squashed Gaussian Policy

در الگوریتم SAC با استفاده از ترفندهای پارامتری‌سازی مجدد، عمل‌ها از یک توزیع نرمال به وسیله نویز تصادفی تولید شده و به این ترتیب امکان محاسبه مشتق‌ها به طور مستقیم از طریقتابع توزیع فراهم می‌شود، که باعث ثبات و کارایی بیشتر در آموزش می‌شود. اما در حالت بدون پارامتری‌سازی مجدد، عمل‌ها مستقیماً از توزیع سیاست نمونه‌برداری می‌شوند و محاسبه گرادیان نیازمند استفاده از ترفندهای احتمال<sup>۵۱</sup> است که عموماً باعث افزایش واریانس و ناپایداری در آموزش می‌شود.

$$\mathbb{E}_{a \sim \pi_\theta} [Q^{\pi_\theta}(s, a) - \alpha \log \pi_\theta(a|s)] = \mathbb{E}_{\xi \sim \mathcal{N}} [Q^{\pi_\theta}(s, \tilde{a}_\theta(s, \xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s, \xi)|s)] \quad (40-4)$$

برای به دست آوردن تابع هزینه سیاست، گام نهایی این است که باید  $Q^{\pi_\theta}$  را با یکی از تخمین‌زننده‌های تابع خود جایگزین کنیم. برخلاف TD3 که از  $Q_{\phi_1}$  ( فقط اولین تخمین‌زننده Q ) استفاده می‌کند، SAC از  $\min_{j=1,2} Q_{\phi_j}$  ( کمینه‌ی دو تخمین‌زننده Q ) استفاده می‌کند. بنابراین، سیاست طبق رابطه زیر بهینه می‌شود:

$$\max_{\theta} \mathbb{E}_{\substack{s \sim \mathcal{D} \\ \xi \sim \mathcal{N}}} \left[ \min_{j=1,2} Q_{\phi_j}(s, \tilde{a}_\theta(s, \xi)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s, \xi)|s) \right] \quad (41-4)$$

که تقریباً مشابه بهینه‌سازی سیاست در DDPG و min-double-Q TD3 است، به جز ترفندهای TD3 و عبارت آنتروپی:

## ۸-۴-۴ اکتشاف و بهره‌برداری در SAC

الگوریتم SAC یک سیاست تصادفی با تنظیم‌سازی آنتروپی آموزش می‌دهد و به صورت سیاست محور به اکتشاف می‌پردازد. ضریب تنظیم آنتروپی  $\alpha$  به طور صریح تعادل بین اکتشاف و بهره‌برداری را کنترل می‌کند، به طوری که مقادیر بالاتر  $\alpha$  به اکتشاف بیشتر و مقادیر پایین‌تر  $\alpha$  به بهره‌برداری بیشتر منجر می‌شود. مقدار بهینه  $\alpha$  ( که به یادگیری پایدارتر و پاداش بالاتر منجر می‌شود ) ممکن است در محیط‌های مختلف متفاوت باشد و نیاز به تنظیم دقیق داشته باشد. در زمان آزمایش، برای ارزیابی میزان بهره‌برداری سیاست از آنچه یاد گرفته است، تصادفی بودن را حذف کرده و از عمل میانگین به جای نمونه‌برداری از توزیع استفاده می‌کنیم. این روش عموماً عملکرد را نسبت به سیاست تصادفی بهبود می‌بخشد.

<sup>51</sup>Likelihood Ratio Trick

## ۹-۴-۴ شبکه SAC

در این بخش الگوریتم SAC پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم <sup>۳</sup> در محیط پایتون با استفاده از کتابخانه PyTorch <sup>[۶۰]</sup> پیاده‌سازی شده است.

### الگوریتم ۳ عامل عملگرد نقاد نرم

وروودی: پارامترهای اولیه سیاست  $(\theta)$ , پارامترهای تابع  $Q(\phi_1, \phi_2)$ , بافر بازی خالی  $(\mathcal{D})$

- ۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید  $\theta_{\text{targ}} \leftarrow \theta$ ,  $\phi_{\text{targ},2} \leftarrow \phi_2$ ,  $\phi_{\text{targ},1} \leftarrow \phi_1$
- ۲: تا وقتی همگرایی رخ دهد:
- ۳: وضعیت  $(s)$  را مشاهده کرده و عمل  $\pi_\theta(\cdot|s) \sim a$  را انتخاب کنید.
- ۴: عمل  $a$  را در محیط اجرا کنید.
- ۵: وضعیت بعدی  $s'$ , پاداش  $r$  و سیگنال پایان  $d$  را مشاهده کنید تا نشان دهد آیا  $s'$  پایانی است یا خیر.
- ۶: اگر  $s'$  پایانی است، وضعیت محیط را بازنگشانی کنید.
- ۷: اگر زمان بهروزرسانی فرا رسیده است:
- ۸: به ازای  $j$  در هر تعداد بهروزرسانی:
- ۹: یک دسته تصادفی گذر از یک حالت به حالت دیگر، از  $\mathcal{D}$ ,  $B = \{(s, a, r, s', d), \dots\}$  از نمونه‌گیری شود.
- ۱۰: هدف را محاسبه کنید:

$$y(r, s', d) = r + \gamma(1 - d) \left( \min_{i=1,2} Q_{\phi_{\text{targ},i}}(s', \tilde{a}') - \alpha \log \pi_\theta(\tilde{a}'|s') \right), \quad \tilde{a}' \sim \pi_\theta(\cdot|s')$$

تابع  $Q$  را با یک مرحله از نزول گرادیان با استفاده از رابطه زیر بهروزرسانی کنید:

$$\nabla_{\phi_i} \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_{\phi_i}(s, a) - y(r, s', d))^2 \quad \text{for } i = 1, 2$$

سیاست را با یک مرحله از صعود گرادیان با استفاده از رابطه زیر بهروزرسانی کنید:

$$\nabla_\theta \frac{1}{|B|} \sum_{s \in B} \left( \min_{i=1,2} Q_{\phi_i}(s, \tilde{a}_\theta(s)) - \alpha \log \pi_\theta(\tilde{a}_\theta(s)|s) \right)$$

شبکه‌های هدف را با استفاده از معادلات زیر بهروزرسانی کنید:

$$\phi_{\text{targ},i} \leftarrow \rho \phi_{\text{targ},i} + (1 - \rho) \phi_i \quad \text{for } i = 1, 2$$

## ۵-۴ عامل بهینه‌سازی سیاست مجاور

الگوریتم بهینه‌سازی سیاست مجاور<sup>۵۲</sup> یک الگوریتم بهینه‌سازی سیاست مبتنی بر گرادیان است که برای حل مسائل کنترل مسئله‌های یادگیری تقویتی استفاده می‌شود. این الگوریتم از الگوریتم TRPO<sup>۵۳</sup> الهام گرفته شده است و با اعمال تغییراتی بر روی آن، سرعت و کارایی آن را افزایش داده است. در این بخش به بررسی این الگوریتم و نحوه عملکرد آن می‌پردازیم. الگوریتم PPO همانند سایر الگوریتم‌های یادگیری تقویتی، به‌دلیل یافتن بهترین گام ممکن برای بهبود عملکرد سیاست با استفاده از داده‌های موجود است. این الگوریتم تلاش می‌کند تا از گام‌های بزرگ که می‌توانند منجر به افت ناگهانی عملکرد شوند، اجتناب کند. برخلاف روش‌های پیچیده‌تر مرتبه دوم مانند TRPO، PPO از مجموعه‌ای از روش‌های مرتبه اول ساده‌تر برای حفظ نزدیکی سیاست‌های جدید به سیاست‌های قبلی استفاده می‌کند. این سادگی در پیاده‌سازی، PPO را به روشی کارآمدتر تبدیل می‌کند، در حالی که از نظر تجربی نشان داده شده است که عملکردی حداقل به اندازه TRPO دارد. از جمله ویژگی‌های مهم این الگوریتم می‌توان به سیاست محور بودن آن اشاره کرد. این الگوریتم برای عامل‌های یادگیری تقویتی که سیاست‌های پیوسته و گسسته دارند، مناسب است.

الگوریتم PPO داری دو گونه اصلی PPO-Clip و PPO-Penalty است. در ادامه به بررسی هر یک از این دو گونه پرداخته شده است.

• روش PPO-Penalty: روش PPO-Penalty به‌دلیل حل تقریبی و بهروزرسانی با محدودیت

واگرایی کولباک-لیبلر<sup>۵۴</sup> است، مشابه روشی که در الگوریتم TRPO استفاده شده است. با این حال، به جای اعمال یک محدودیت سخت<sup>۵۵</sup>، PPO-Penalty واگرایی KL را در تابع هدف جرمیه می‌کند. این جرمیه به طور خودکار در طول آموزش تنظیم می‌شود تا از افت ناگهانی عملکرد جلوگیری کند.

• روش PPO-Clip: در این روش، هیچ عبارت واگرایی KL در تابع هدف وجود ندارد و هیچ محدودیتی

اعمال نمی‌شود. در عوض، در PPO-Clip از یک عملیات بریدن<sup>۵۶</sup> خاص در تابع هدف استفاده می‌کند تا انگیزه سیاست جدید برای دور شدن از سیاست قبلی را از بین ببرد.

در این پژوهش از روش PPO-Clip برای آموزش عامل‌های یادگیری تقویتی استفاده شده است.

<sup>52</sup>Proximal Policy Optimization (PPO)

<sup>53</sup>Trust Region Policy Optimization

<sup>54</sup>Kullback-Leibler (KL) Divergence

<sup>55</sup>Hard Constraint

<sup>56</sup>Clipping

## ۱-۵-۴ سیاست در الگوریتم PPO

تابع سیاست در الگوریتم PPO به صورت یک شبکه عصبی پیاده‌سازی شده است. این شبکه عصبی ورودی‌های محیط را دریافت کرده و اقدامی را که باید عامل انجام دهد را تولید می‌کند. این شبکه عصبی می‌تواند شامل چندین لایه پنهان با توابع فعال‌سازی مختلف باشد. در این پژوهش از یک شبکه عصبی با سه لایه پنهان و تابع فعال‌سازی ReLu استفاده شده است. تابع سیاست در الگوریتم PPO به صورت زیر بهروزرسانی می‌شود:

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{s,a \sim \pi_{\theta_k}} [L(s,a,\theta_k, \theta)] \quad (42-4)$$

در این پژوهش برای به حداقل رساندن تابع هدف، چندین گام بهینه‌سازی گرادیان کاهشی تصادفی<sup>۵۷</sup> اجرا شده است. در معادله بالا  $L$  به صورت زیر تعریف شده است:

$$L(s,a,\theta_k, \theta) = \min \left( \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), \text{clip} \left( \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}, 1-\epsilon, 1+\epsilon \right) A^{\pi_{\theta_k}}(s,a) \right) \quad (43-4)$$

که در آن  $\epsilon$  یک ابرپارامتر است که مقدار آن معمولاً کوچک است. این ابرپارامتر مشخص می‌کند که چقدر اندازه گام بهینه‌سازی باید محدود شود. در این پژوهش مقدار  $0.2 = \epsilon$  انتخاب شده است. جهت سادگی در پیاده‌سازی معادله (43-4) به معادله تغیر داده شده است.

$$L(s,a,\theta_k, \theta) = \min \left( \frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s,a), g(\epsilon, A^{\pi_{\theta_k}}(s,a)) \right) \quad (44-4)$$

که تابع  $g$  به صورت زیر تعریف شده است.

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0 \end{cases} \quad (45-4)$$

در حالی که این نوع محدود کردن (PPO-Clip) تا حد زیادی به اطمینان از بهروزرسانی‌های معقول سیاست کمک می‌کند، همچنان ممکن است سیاستی به دست آید که بیش از حد از سیاست قدیمی دور باشد. برای جلوگیری از این امر، پیاده‌سازی‌های مختلف PPO از مجموعه‌ای از ترفندها استفاده می‌کنند. در پیاده‌سازی این پژوهش، از روشی ساده به نام توقف زودهنگام<sup>۵۸</sup> استفاده شده است. اگر میانگین واگرایی کولباک-لیبلر (KL) خط‌مشی جدید از خط‌مشی قدیمی از یک آستانه فراتر رود، گام‌های گرادیان (بهینه‌سازی) را متوقف می‌شوند.

<sup>57</sup>Stochastic Gradient Descent (SGD)

<sup>58</sup>Early Stopping

## ۲-۵-۴ اکتشاف و بهره‌برداری در PPO

الگوریتم PPO از یک سیاست تصادفی به صورت سیاست‌محور برای آموزش استفاده می‌کند. این به این معنی است که اکتشاف محیط با نمونه‌گیری عمل‌ها بر اساس آخرین نسخه از این سیاست تصادفی انجام می‌شود. میزان تصادفی بودن انتخاب عمل به شرایط اولیه و فرآیند آموزش بستگی دارد.

در طول آموزش، سیاست به طور کلی به تدریج کمتر تصادفی می‌شود، زیرا قانون بهروزرسانی آن را تشویق می‌کند تا از پاداش‌هایی که قبلاً پیدا کرده است، بهره‌برداری کند. البته این موضوع می‌تواند منجر به رسیدن سیاست به بهینه‌های محلی<sup>۵۹</sup> شود.

## ۳-۵-۴ شبکه PPO

در این بخش الگوریتم PPO پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم<sup>۴</sup> در محیط پایتون با استفاده از کتابخانه PyTorch [۶۰] پیاده‌سازی شده است.

---

<sup>59</sup>Local Optima

---

#### الگوریتم ۴ بهینه‌سازی سیاست مجاور (PPO-Clip)

---

وروودی: پارامترهای اولیه سیاست  $(\theta_0)$ ، پارامترهای تابع ارزش  $(\phi_0)$

۱: به ازای ...  $k = 0, 1, 2, \dots$

۲: مجموعه‌ای از مسیرها به نام  $\{\tau_i\}$  با اجرای سیاست  $\pi(\theta_k) = \pi_k$  در محیط جمع‌آوری شود.

۳: پاداش‌های باقی‌مانده  $(\hat{R}_t)$  محاسبه شود.

۴: برآوردهای مزیت را محاسبه کنید،  $\hat{A}_t$  (با استفاده از هر روش تخمین مزیت) بر اساس تابع ارزش

فعالی  $.V_{\phi_k}$

۵: سیاست را با به حداکثر رساندن تابع هدف PPO-Clip به روزرسانی کنید:

$$\theta_{k+1} = \arg \max_{\theta} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \min \left( \frac{\pi_{\theta}(a_t | s_t)}{\pi_{\theta_k}(a_t | s_t)} A^{\pi_{\theta_k}}(s_t, a_t), g(\epsilon, A^{\pi_{\theta_k}}(s_t, a_t)) \right)$$

۶: معمولاً از طریق گرادیان افزایشی تصادفی Adam. برازش تابع ارزش با رگرسیون بر روی میانگین مربعات خطای

$$\phi_{k+1} = \arg \min_{\phi} \frac{1}{|\mathcal{D}_k|T} \sum_{\tau \in \mathcal{D}_k} \sum_{t=0}^T \left( V_{\phi}(s_t) - \hat{R}_t \right)^2$$

ممولاً از طریق برخی از الگوریتم‌های کاهشی گرادیان.

---

## فصل ۵

### شبیه‌سازی عامل در محیط سه جسمی

در این فصل، فرآیند شبیه‌سازی عامل هوشمند کنترل‌کننده فضای‌پیما در محیط دینامیکی سه جسمی بررسی شده است. در بخش ۱-۵ به طراحی و در بخش ۲-۵ به شبیه‌سازی عامل هدایت‌کننده مبتنی بر یادگیری تقویتی است پرداخته شده است. این عامل طراحی و شبیه‌سازی شده باید توانایی این را داشته باشد که فضای‌پیما را به طور مؤثر به سمت اهداف تعیین‌شده هدایت کند، در حالی که محدودیت‌هایی نظری مصرف سوخت و وجود اغتشاش دارد.

#### ۱-۵ طراحی عامل

در این زیربخش، معماری عامل هوشمند کنترل‌کننده فضای‌پیما در محیط سه‌جسمی شرح داده شده است. این معماری شامل تعریف فضای حالت، عمل و تابع پاداش است.

#### ۱-۱-۵ فضای حالت

فضای حالت<sup>۱</sup> در این پژوهش به گونه‌ای طراحی شده است که وضعیت دینامیکی فضای‌پیما را نسبت به یک مسیر و سرعت مرجع مشخص می‌کند. این فضا شامل اختلاف‌های موقعیت و سرعت از مسیر و سرعت مرجع است و به صورت زیر تعریف می‌شود:

$$S = \{\delta x, \delta y, \delta \dot{x}, \delta \dot{y}\}$$

که در آن:

---

<sup>1</sup>State Space

•  $\delta x, \delta y$ : اختلاف موقعیت فضاییما نسبت به مسیر مرجع در محورهای  $x, y$ .

•  $\dot{\delta}x, \dot{\delta}y$ : اختلاف سرعت فضاییما نسبت به سرعت مرجع در محورهای  $x, y$ .

هر یک از این متغیرها به طور مستقل وضعیت فضاییما را در یک جهت خاص توصیف می‌کنند و امکان تحلیل دقیق انحرافات را فراهم می‌سازند. استفاده از اختلافهای موقعیت و سرعت به جای مقادیر مطلق، به دلایل زیر انجام شده است:

• **تمرکز بر انحرافات:** هدف اصلی سیستم کنترلی، کاهش انحرافات از مسیر و سرعت مطلوب است. با استفاده از اختلاف‌ها، کنترلر می‌تواند به طور مستقیم بر این انحرافات اثر بگذارد و نیازی به محاسبه مقادیر مطلق موقعیت و سرعت ندارد.

• **سازگاری با یادگیری تقویتی:** در الگوریتم‌های یادگیری تقویتی، فضاهای حالت مبتنی بر اختلاف معمولاً دامنه محدودتری دارند که فرآیند یادگیری را سریع‌تر و پایدارتر می‌کند.

## ۲-۱-۵ فضای عمل

فضای عمل<sup>۲</sup> فضاییما با پیشران کم مجموعه‌ای از عمل‌های پیوسته است که فضاییما می‌تواند در محیط شبیه‌سازی انجام دهد. این فضا به گونه‌ای طراحی شده که امکان اعمال نیرو در جهت‌های مشخص و با مقادیر مناسب با توان واقعی فضاییماها فراهم شود. به طور خاص، فضای اقدام شامل موارد زیر است:

• **نیروی اعمال شده در جهت  $x$ :** این متغیر پیوسته، مقدار نیرویی را که در جهت محور  $x$  به فضاییما وارد می‌شود، تعیین می‌کند. دامنه این نیرو بر اساس توان پیشرانه‌های موجود در فضاییماهای واقعی انتخاب شده است. به عبارت دیگر، اگر حداقل نیروی قابل اعمال در جهت  $x$  برابر با  $f_{x,\max}$  باشد، این متغیر می‌تواند مقادیری در بازه  $[-f_{x,\max}, f_{x,\max}]$  داشته باشد.

• **نیروی اعمال شده در جهت  $y$ :** این متغیر پیوسته، مقدار نیرویی را که در جهت محور  $y$  به فضاییما وارد می‌شود، مشخص می‌کند. مشابه جهت  $x$ ، دامنه این نیرو نیز بر اساس توان پیشرانه‌های موجود تعیین شده و می‌تواند در بازه  $[-f_{y,\max}, f_{y,\max}]$  قرار گیرد.

انتخاب این نیروها بر اساس ویژگی‌های واقعی فضاییماها، به ویژه توان و محدودیت‌های پیشرانه‌های آنها، صورت گرفته است. این امر اطمینان می‌دهد که شبیه‌سازی تا حد ممکن به شرایط واقعی نزدیک باشد و نتایج

<sup>2</sup>Action Space

به دست آمده قابلیت تعمیم به کاربردهای عملی را داشته باشند. همچنین، تعریف فضای اقدام به صورت پیوسته، امکان کنترل دقیق و انعطاف‌پذیر بر حرکت فضاییما را فراهم می‌کند، که برای دستیابی به اهداف کنترلی در محیط‌های دینامیکی پیچیده ضروری است. به طور خلاصه، فضای اقدام به صورت زیر تعریف می‌شود:

$$a = \{f_x, f_y \mid f_x \in [-f_{x,\max}, f_{x,\max}], f_y \in [-f_{y,\max}, f_{y,\max}]\}$$

### ۳-۱-۵ تابع پاداش

تابع پاداش<sup>۳</sup> به منظور هدایت رفتار عامل طراحی شده و شامل دو مؤلفه اصلی است:

- پاداش برای دستیابی به هدف: تشویق عامل برای نزدیک شدن به مدار هدف.
- جریمه برای مصرف سوخت: تنبیه برای استفاده بیش از حد از پیشرانه.
- جریمه برای انحراف از مسیر مرجع: تنبیه برای خروج از مسیر مرجع.

تابع پاداش به صورت زیر تعریف می‌شود:

$$r(s, a) = r_{\text{target}}(s) + r_{\text{thrust}}(a) + r_{\text{divergence}}(s)$$

که در آن مؤلفه‌های تابع پاداش به صورت زیر تعریف شده‌اند:

$$r_{\text{target}}(s) = -k_1 \cdot d(s, s_{\text{target}}) \quad (1-5)$$

$$r_{\text{thrust}}(a) = -k_2 \cdot |0a|0 \quad (2-5)$$

$$r_{\text{divergence}}(s) = \begin{cases} -k_3 & \text{if } d(s, s_{\text{reference}}) > \epsilon \\ 0 & \text{otherwise} \end{cases} \quad (3-5)$$

تابع  $d(s, s')$  فاصله بین دو وضعیت  $s$  و  $s'$  را نشان می‌دهد که معمولاً به صورت فاصله اقلیدسی محاسبه می‌شود. ضرایب  $k_1, k_2, k_3$  از طریق آزمایش و خطای تنظیم شده‌اند تا تعادل مناسبی بین دستیابی به هدف، بهینه‌سازی مصرف سوخت، و حفظ مسیر مرجع برقرار شود. علاوه بر این، این ضرایب تأثیر مستقیمی بر پایداری و فرآیند یادگیری عامل دارند. به عنوان مثال، انتخاب مقادیر بیش از حد بزرگ برای  $k_1$  ممکن است باعث شود عامل به سرعت به سمت هدف حرکت کند اما پایداری مسیر را از دست بدهد، در حالی که مقادیر بزرگ  $k_3$  می‌توانند عامل را بیش از حد محافظه‌کار کرده و فرآیند یادگیری را کند نمایند. تنظیم دقیق این ضرایب، نه تنها عملکرد عامل را بهینه می‌کند، بلکه پایداری عددی و سرعت همگرایی الگوریتم یادگیری تقویتی را نیز تضمین می‌نماید.

---

<sup>3</sup>Reward Function

## ۲-۵ شبیه‌سازی عامل

در این زیربخش، فرآیند شبیه‌سازی و آموزش عامل با استفاده از الگوریتم‌های یادگیری تقویتی پیش‌رفته شرح داده می‌شود. الگوریتم‌های مورد استفاده، مراحل آموزش، و نتایج حاصل از شبیه‌سازی ارائه می‌گردند.

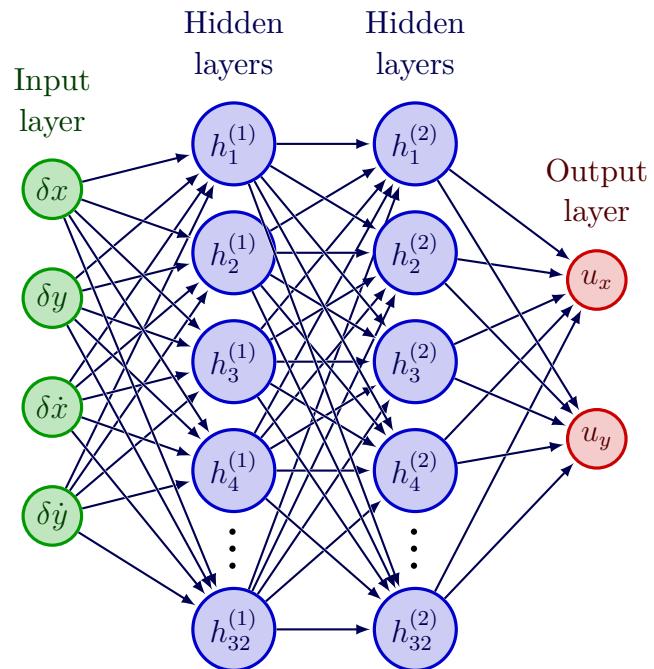
### ۱-۲-۵ پارامترهای یادگیری الگوریتم‌های مورد استفاده

برای آموزش عامل، الگوریتم‌های زیر به کار گرفته شده‌اند:

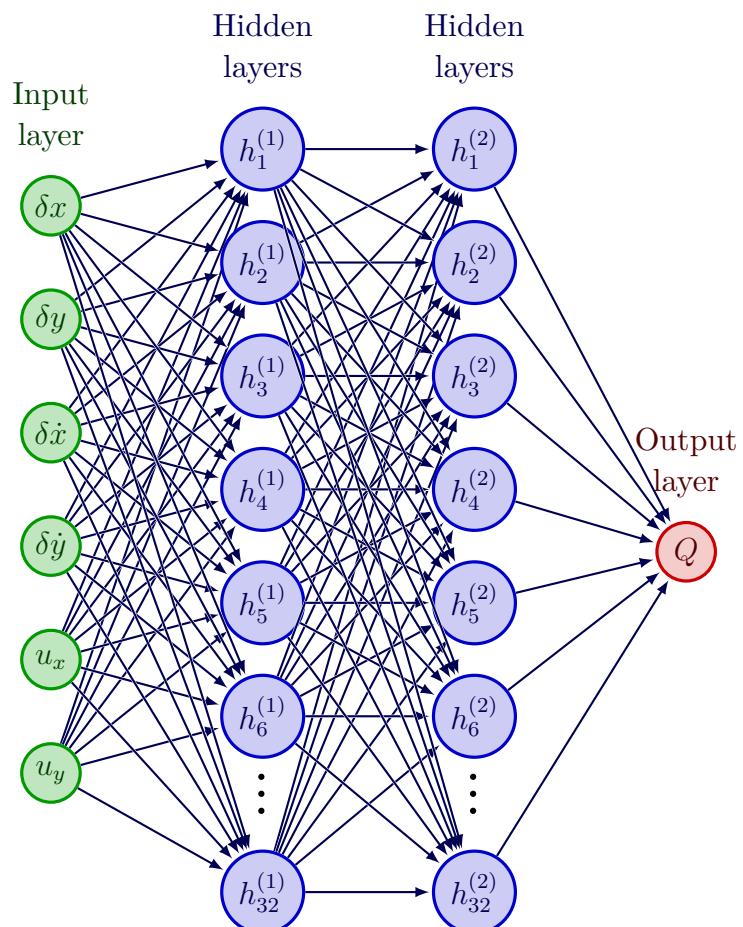
جدول ۱-۵: ویژگی‌های الگوریتم‌های مورد استفاده در شبیه‌سازی

تعداد پارامترها	شبکه Critic		شبکه Actor		الگوریتم
	نودها	لایه‌ها	نودها	لایه‌ها	
$150 \times 10^3$	$(2^8, 2^5)$	3	$(2^8, 2^7, 2^6)$	3	DDPG
$50 \times 10^3$	$(2^7, 2^6)$	2	$(2^7, 2^6)$	2	PPO
$160 \times 10^3$	$(2^8, 2^7, 2^6)$	3	$(2^8, 2^7, 2^6)$	3	SAC
$200 \times 10^3$	$(2^8, 2^7, 2^7, 2^6)$	4	$(2^8, 2^7, 2^6)$	3	TD3

این الگوریتم‌ها به دلیل توانایی در مدیریت فضاهای پیوسته و عملکرد مؤثر در محیط‌های پیچیده انتخاب شده‌اند. در شکل‌های ۱-۵ و ۲-۵ ساختار شبیه‌سازی شده شبکه عصبی عامل و نقاد آورده شده است.



شكل ١-٥: ساختار شبکه عصبی عامل



شكل ٢-٥: ساختار شبکه عصبی نقاد

نام پارامتر	مقدار	نام پارامتر	مقدار
تعداد دوره‌های یادگیری	100	گام در هر دوره یادگیری	
ضریب تنزیل ( $\gamma$ )	0.99	اندازه‌ی مخزن تجربه	
نرخ یادگیری سیاست	$10^{-3}$	ضریب میانگین پلیاک	
اندازه‌ی دسته	1024	نرخ یادگیری Q	
گام شروع به روزرسانی	1 000	گام شروع استفاده از سیاست	
نویز عمل	0.1	فاصله‌ی به روزرسانی	
دستگاه	Cuda	حداکثر طول رخداد	
تابع فعال‌سازی Actor	ReLU	اندازه شبکه‌ی Actor	
تابع فعال‌سازی Critic	ReLU	اندازه شبکه‌ی Critic	

جدول ۵-۲: جدول پارامترها و مقادیر پیش‌فرض الگوریتم DDPG [۱]

نام پارامتر	مقدار	نام پارامتر	مقدار
تعداد دوره‌های یادگیری	100	گام در هر دوره یادگیری	
ضریب تنزیل ( $\gamma$ )	0.99	اندازه‌ی مخزن تجربه	
نرخ یادگیری سیاست	$10^{-3}$	ضریب میانگین پلیاک	
اندازه‌ی دسته	1024	نرخ یادگیری Q	
گام شروع به روزرسانی	1 000	گام شروع استفاده از سیاست	
نویز عمل	0.1	فاصله‌ی به روزرسانی	
برش نویز	0.5	نویز هدف	
حداکثر طول رخداد	30 000	تأثیر در به روزرسانی سیاست	
تابع فعال‌سازی Actor	ReLU	اندازه شبکه‌ی Actor	
تابع فعال‌سازی Critic	ReLU	اندازه شبکه‌ی Critic	

جدول ۳-۵: جدول پارامترها و مقادیر پیش‌فرض الگوریتم TD3 [۱]

نام پارامتر	مقدار	نام پارامتر	مقدار
تعداد دوره‌های یادگیری	30 000	گام در هر دوره یادگیری	100
ضریب تنزیل ( $\gamma$ )	$10^6$	اندازه‌ی مخزن تجربه	0.99
نرخ یادگیری	0.995	ضریب میانگین پلیاک	$10^{-3}$
اندازه‌ی دسته	0.2	نرخ دمای آلفا	1024
گام شروع بهروزرسانی	5 000	گام شروع استفاده از سیاست	1 000
فاصله‌ی بهروزرسانی	10	تعداد بهروزرسانی در هر مرحله	2 000
حداکثر طول رخداد	10	تعداد اپیزودهای آزمون	30 000
تابع فعال‌سازی Actor	$(2^5, 2^5)$	اندازه شبکه‌ی Actor	ReLU
تابع فعال‌سازی Critic	$(2^5, 2^5)$	اندازه شبکه‌ی Critic	ReLU

جدول ۴-۵: جدول پارامترها و مقادیر پیش‌فرض الگوریتم SAC [۱]

نام پارامتر	مقدار	نام پارامتر	مقدار
تعداد دوره‌های یادگیری	30 000	گام در هر دوره یادگیری	100
ضریب برش ratio clip	0.99	ضریب تنزیل ( $\gamma$ )	0.2
نرخ یادگیری تابع ارزش	$3 \times 10^{-4}$	نرخ یادگیری سیاست	$10^{-3}$
تعداد تکرار آموزش ارزش	80	تعداد تکرار آموزش سیاست	80
تابع فعال‌سازی Actor	$(2^5, 2^5)$	اندازه شبکه‌ی Actor	ReLU
تابع فعال‌سازی Critic	$(2^5, 2^5)$	اندازه شبکه‌ی Critic	ReLU

جدول ۵-۵: جدول پارامترها و مقادیر پیش‌فرض الگوریتم PPO [۱]

## ۲-۶-۵ فرآیند آموزش

آموزش عامل به صورت کلی در چند مرحله انجام شده است. ابتدا، کاوش اولیه در محیط با استفاده از یک سیاست تصادفی صورت گرفته و تجربه‌های اولیه جمع‌آوری شده‌اند. سپس، شبکه‌های عصبی الگوریتم‌ها با بهره‌گیری از این تجربه‌ها بهروزرسانی شده‌اند. در نهایت، پارامترهای کلیدی مانند نرخ یادگیری و اندازه بافر تجربه تنظیم شده‌اند تا پایداری فرآیند تضمین شود.

برای پیاده‌سازی این فرآیند، از چارچوب PyTorch استفاده شده است. همچنین، به منظور جلوگیری از بیش‌برازش، تکنیک Exploration Noise به کار گرفته شده است. آموزش تا زمانی ادامه یافته که موفقیت عامل در بیش از ۹۰ درصد موارد به دست آمده باشد. در این راستا، برای بهینه‌سازی پارامترهای شبکه‌های

عصبی، از روش Backpropagation استفاده شده است. این روش بر اساس گرادیان تابع خطا نسبت به پارامترها عمل می‌کند که به صورت زیر بیان می‌شود:

$$\frac{\partial L}{\partial w} = \frac{\partial L}{\partial y} \cdot \frac{\partial y}{\partial w} \quad (4-5)$$

که در آن  $L$  تابع خطا،  $w$  وزن‌های شبکه، و  $y$  خروجی شبکه عصبی است. به روزرسانی وزن‌ها با استفاده از روش گرادیان نزولی انجام شده است:

$$w_{t+1} = w_t - \eta \cdot \frac{\partial L}{\partial w} \quad (5-5)$$

که  $\eta$  نرخ یادگیری است و به عنوان یک پارامتر کلیدی تنظیم شده است.

## فصل ۶

# یادگیری تقویتی چندعاملی

کاربردهای پیچیده در یادگیری تقویتی نیازمند اضافه کردن چندین عامل<sup>۱</sup> برای انجام همزمان وظایف مختلف هستند. با این حال، افزایش تعداد عامل‌ها چالش‌هایی در مدیریت تعاملات میان آن‌ها به همراه دارد. در این فصل، بر اساس مسئله بهینه‌سازی برای هر عامل، مفهوم تعادل<sup>۲</sup> معرفی شده تا رفتارهای توزیعی چندعاملی را تنظیم کند. رابطه رقابت میان عامل‌ها در سناریوهای مختلف تحلیل شده و آن‌ها با الگوریتم‌های معمول یادگیری تقویتی چندعاملی ترکیب شده‌اند. بر اساس انواع تعاملات، یک چارچوب نظریه بازی برای مدل‌سازی عمومی در سناریوهای چندعاملی استفاده شده است. با تحلیل بهینه‌سازی و وضعیت تعادل برای هر بخش از چارچوب، سیاست بهینه یادگیری تقویتی چندعاملی برای هر عامل بررسی شده است.

## ۱-۶ تعاریف و مفاهیم اساسی

یادگیری تقویتی چندعاملی<sup>۳</sup> به بررسی چگونگی یادگیری و تصمیم‌گیری چندین عامل مستقل در یک محیط مشترک پرداخته می‌شود. برای تحلیل دقیق و درک بهتر این حوزه، اجزای اصلی آن شامل عامل، سیاست و مطلوبیت<sup>۴</sup> در نظر گرفته می‌شوند که در ادامه به صورت مختصر و منسجم تشریح می‌گردند.

- عامل: یک موجودیت مستقل به عنوان عامل تعریف می‌شود که به صورت خودمختار با محیط تعامل کرده و بر اساس مشاهدات رفتار سایر عامل‌ها، سیاست‌هاییش انتخاب می‌گردد تا سود حداکثر یا ضرر حداقل حاصل شود. در سناریوهای مورد بررسی، چندین عامل به صورت مستقل عمل می‌کنند؛ اما اگر

<sup>1</sup>Multi-Agent

<sup>2</sup>Equilibrium

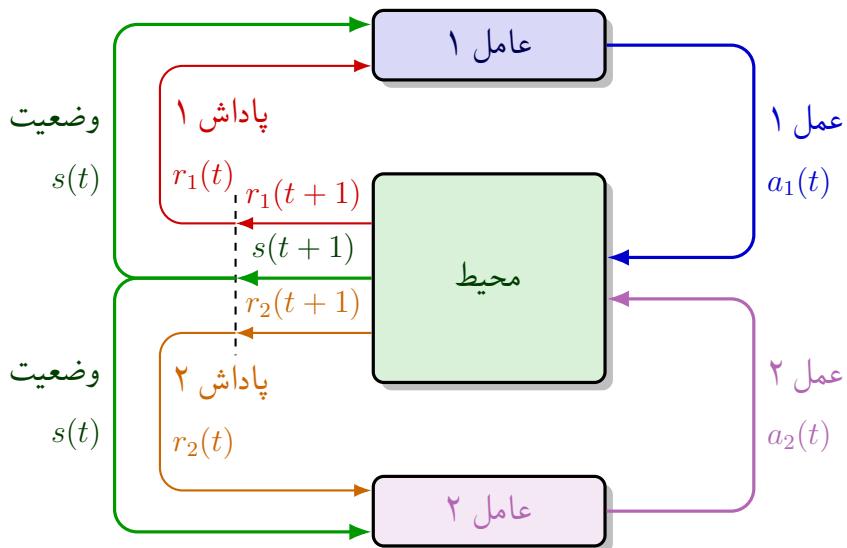
<sup>3</sup>Multi-Agent Reinforcement Learning (MARL)

<sup>4</sup>Utility

تعداد عامل‌ها به یک کاوش یابد، MARL به یادگیری تقویتی معمولی تبدیل می‌شود.

- سیاست: برای هر عامل در MARL، سیاستی خاص در نظر گرفته می‌شود که به عنوان روشی برای انتخاب اقدامات بر اساس وضعیت محیط و رفتار سایر عامل‌ها تعریف می‌گردد. این سیاست‌ها با هدف به حداقل رساندن سود و به حداقل رساندن هزینه طراحی شده و تحت تأثیر محیط و سیاست‌های دیگر عامل‌ها قرار می‌گیرند.
- مطلوبیت: مطلوبیت هر عامل بر اساس نیازها و وابستگی‌هایش به محیط و سایر عامل‌ها تعریف شده و به صورت سود منهای هزینه، با توجه به اهداف مختلف محاسبه می‌شود. در سناریوهای چندعاملی، از طریق یادگیری از محیط و تعامل با دیگران، مطلوبیت هر عامل بهینه می‌گردد.

در این چارچوب، برای هر عامل در MARL تابع مطلوبیت خاصی در نظر گرفته شده و بر اساس مشاهدات و تجربیات حاصل از تعاملات، یادگیری سیاست به صورت مستقل انجام می‌شود تا ارزش مطلوبیت به حداقل برسد، بدون اینکه مستقیماً به مطلوبیت سایر عامل‌ها توجه شود. این فرآیند ممکن است به رقابت یا همکاری میان عامل‌ها منجر گردد. با توجه به پیچیدگی تعاملات میان چندین عامل، تحلیل نظریه بازی‌ها به عنوان ابزاری مؤثر برای تصمیم‌گیری در این حوزه به کار گرفته می‌شود. بسته به سناریوهای مختلف، این بازی‌ها در دسته‌بندی‌های متفاوتی قرار داده شده که در بخش‌های بعدی بررسی خواهند شد.



شکل ۱-۶: حلقه تعامل عامل‌های یادگیری تقویتی چند عاملی با محیط

## ۲-۶ نظریه بازی‌ها

نظریه بازی‌ها شاخه‌ای از ریاضیات است که به مطالعه تصمیم‌گیری در موقعیت‌هایی می‌پردازد که نتیجه انتخاب‌های هر فرد به تصمیمات دیگران وابسته است. این نظریه چارچوبی برای تحلیل تعاملات میان بازیکنان ارائه می‌دهد و در حوزه‌های مختلفی مانند اقتصاد، علوم سیاسی، زیست‌شناسی و علوم کامپیوتر کاربرد دارد. در این فصل، دو مفهوم کلیدی نظریه بازی‌ها یعنی تعادل نش و بازی‌های مجموع صفر بررسی شده است.

### ۱-۲-۶ تعادل نش

تعادل نش<sup>۵</sup> یکی از بنیادی‌ترین مفاهیم در نظریه بازی‌ها است که توسط جان نش در سال ۱۹۵۰ معرفی شد. این مفهوم به مجموعه‌ای از بازی‌ها اشاره دارد که در آن هیچ بازیکنی نمی‌تواند با تغییر یک جانبه سیاست خود، سود بیشتری به دست آورد، به شرطی که سیاست‌های سایر بازیکنان ثابت بماند.

- تعریف تعادل نش: فرض کنید یک بازی با  $n$  بازیکن داریم. هر بازیکن  $i$  دارای مجموعه سیاست‌های  $\Pi_i$  و تابع مطلوبیت  $\mathbb{R} \rightarrow \Pi_1 \times \Pi_2 \times \dots \times \Pi_n : u_i$  است. یک مجموعه سیاست  $(\pi_1^*, \pi_2^*, \dots, \pi_n^*)$  تعادل نش نامیده می‌شود اگر برای هر بازیکن  $i$  و هر سیاست  $\pi_i \in \Pi_i$  در وضعیت  $s$  داشته باشیم:

$$u_i(\pi_i^*, \pi_{-i}^*, s) \geq u_i(\pi_i, \pi_{-i}^*, s) \quad (1-6)$$

در اینجا،  $\pi_{-i}^*$  نشان‌دهنده سیاست‌های همه بازیکنان به جز بازیکن  $i$  است. در ادامه پژوهش جهت استفاده از چارچوب نظریه بازی در یادگیری تقویتی تابع مطلوبیت به‌گونه‌ای تعریف شده است که برابر با تابع ارزش  $V_i^{\pi_i, \pi_{-i}}(s) = V_i(\pi_i, \pi_{-i}, s)$  باشد.

- اهمیت تعادل نش: تعادل نش نقطه‌ای را در بازی مشخص می‌کند که هر بازیکن بهترین پاسخ را نسبت به انتخاب‌های دیگران ارائه داده است. این مفهوم بهویژه در بازی‌های غیرهمکارانه، به عنوان پیش‌بینی رفتار منطقی بازیکنان استفاده می‌شود و در زمینه‌هایی مانند یادگیری تقویتی چند عامله کاربرد گسترده‌ای دارد.

<sup>5</sup>Nash Equilibrium

## ۲-۶ بازی مجموع صفر

بازی‌های مجموع صفر<sup>۶</sup> دسته‌ای از بازی‌ها هستند که در آن‌ها تابع ارزش یک بازیکن دقیقاً برابر با ضرر بازیکن دیگر است؛ بنابراین، مجموع ارزش‌های همه بازیکنان در هر مرحله صفر خواهد بود.

- **تعريف بازی مجموع صفر:**

در یک بازی دو نفره، اگر تابع ارزشِ حالت (value) بازیکن اول  $V_1^{(\pi_1, \pi_2)}(s)$  و بازیکن دوم  $V_2^{(\pi_1, \pi_2)}(s)$  به گونه‌ای باشند که:

$$V_1^{(\pi_1, \pi_2)}(s) + V_2^{(\pi_1, \pi_2)}(s) = 0 \implies V_1^{(\pi_1, \pi_2)}(s) = -V_2^{(\pi_1, \pi_2)}(s), \quad (2-6)$$

آنگاه آن بازی را بازی مجموع صفر می‌نامیم.

به طور مشابه، اگر تابع ارزش-عمل برای دو بازیکن را با  $Q_1^{(\pi_1, \pi_2)}(s, a_1, a_2)$  و  $Q_2^{(\pi_1, \pi_2)}(s, a_1, a_2)$  نشان دهیم، باید برقرار باشد:

(3-6)

$$Q_1^{(\pi_1, \pi_2)}(s, a_1, a_2) + Q_2^{(\pi_1, \pi_2)}(s, a_1, a_2) = 0 \implies Q_1^{(\pi_1, \pi_2)}(s, a_1, a_2) = -Q_2^{(\pi_1, \pi_2)}(s, a_1, a_2).$$

- **سیاست بهینه در بازی مجموع صفر:**

در این بازی‌ها، هر بازیکن سیاستی را برمی‌گزیند که تابع ارزش خود را در برابر بهترین پاسخِ حریف بیشینه کند؛ این انتخاب در نهایت به تعادل نش منجر می‌شود.

به صورت تابع ارزشِ حالت:

$$V_1^*(s) = \max_{\pi_1} \min_{\pi_2} V_1^{(\pi_1, \pi_2)}(s), \quad (4-6)$$

$$V_2^*(s) = \max_{\pi_2} \min_{\pi_1} V_2^{(\pi_1, \pi_2)}(s). \quad (5-6)$$

و به صورت تابع ارزش-عمل:

$$Q_1^*(s, a_1, a_2) = \max_{\pi_1} \min_{\pi_2} Q_1^{(\pi_1, \pi_2)}(s, a_1, a_2), \quad (6-6)$$

$$Q_2^*(s, a_1, a_2) = \max_{\pi_2} \min_{\pi_1} Q_2^{(\pi_1, \pi_2)}(s, a_1, a_2). \quad (7-6)$$

---

<sup>6</sup>Zero-Sum Games

## ۳-۶ گرادیان سیاست عمیق قطعی دو عاملی

گرادیان سیاست عمیق قطعی چند عاملی<sup>۵</sup> توسعه‌ای از الگوریتم DDPG برای محیط‌های چند عاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های دو عاملی مجموع صفر می‌پردازیم که در آن مجموع پاداش‌های دو عامل همواره صفر است (آنچه یک عامل به دست می‌آورد، عامل دیگر از دست می‌دهد).

### ۱-۳-۶ چالش‌های یادگیری تقویتی در محیط‌های چند عاملی

در محیط‌های چند عاملی، سیاست هر عامل مدام در حال تغییر است، که باعث می‌شود محیط از دید هر عامل غیرایستا<sup>۶</sup> شود. این مسئله چالش بزرگی برای الگوریتم‌های یادگیری تقویتی تک عاملی مانند DDPG ایجاد می‌کند، زیرا فرض ایستایی محیط را نقض می‌کند.

MA-DDPG با استفاده از رویکرد آموزش متتمرکز، اجرای غیرمتتمرکز<sup>۷</sup> این مشکل را حل می‌کند. در این رویکرد، هر عامل در زمان آموزش به اطلاعات کامل محیط دسترسی دارد، اما در زمان اجرا تنها از مشاهدات محلی خود استفاده می‌کند.

### ۲-۳-۶ معماری MA-DDPG در بازی‌های مجموع صفر

در یک بازی دو عاملی مجموع صفر، دو عامل با نمادهای ۱ و ۲ نشان داده می‌شوند. هر عامل دارای شبکه‌های منحصر به فرد خود است:

- شبکه‌های بازیگر:  $\mu_{\theta_1}(o_1)$  و  $\mu_{\theta_2}(o_2)$  که مشاهدات محلی  $o_1$  و  $o_2$  را به اعمال  $a_1$  و  $a_2$  نگاشت می‌کنند.
- شبکه‌های منتقد:  $Q_{\phi_2}(o_2, a_2, a_1)$  و  $Q_{\phi_1}(o_1, a_1, a_2)$  که ارزش حالت-عمل را با توجه به مشاهدات و اعمال تمام عامل‌ها تخمین می‌زنند.
- شبکه‌های هدف: مشابه DDPG، برای پایدار کردن آموزش از شبکه‌های هدف استفاده می‌شود.

در بازی‌های مجموع صفر، پاداش‌ها رابطه  $r_1 + r_2 = 0$  دارند که در آن  $r_1$  و  $r_2$  پاداش‌های دریافتی عامل‌ها هستند. در نتیجه،  $-r_1 = r_2$  است که نمایانگر تضاد کامل منافع بین عامل‌هاست.

<sup>7</sup>Multi-Agent Deep Deterministic Policy Gradient (MA-DDPG)

<sup>8</sup>Non-stationary

<sup>9</sup>Centralized Training, Decentralized Execution

### ۳-۳-۶ آموزش MA-DDPG در بازی‌های مجموع‌صفر

فرایند آموزش MA-DDPG برای بازی‌های مجموع‌صفر به شرح زیر است:

#### یادگیری تابع $Q$

برای هر عامل  $i \in \{1, 2\}$ ، تابع  $Q$  با کمینه کردن خطای میانگین مربعات بلمن به روزرسانی می‌شود:

$$L(\phi_i, \mathcal{D}) = \underset{(o, a, r_i, o', d) \sim \mathcal{D}}{\text{E}} \left[ \left( Q_{\phi_i}(o_i, a_1, a_2) - y_i \right)^2 \right] \quad (8-6)$$

که در آن  $(o, a) = (a_1, a_2)$  بردار مشاهدات،  $y_i$  هدف برای عامل  $i$  است:

$$y_i = r_i + \gamma(1 - d)Q_{\phi_i, \text{targ}}(o'_i, \mu_{\theta_1, \text{targ}}(o'_1), \mu_{\theta_2, \text{targ}}(o'_2)) \quad (9-6)$$

توجه کنید که منتقد هر عامل به اعمال همه عامل‌ها دسترسی دارد، اما در بازی‌های مجموع‌صفر، عامل شماره ۲ جهت مخالف هدف عامل ۱ را دنبال می‌کند.

#### یادگیری سیاست

سیاست هر عامل با بیشینه کردن تابع  $Q$  مربوط به آن عامل به روزرسانی می‌شود:

$$\max_{\theta_i} \underset{o \sim \mathcal{D}}{\text{E}} [Q_{\phi_i}(o_i, \mu_{\theta_i}(o_i), \mu_{\theta_{-i}}(o_{-i}))] \quad (10-6)$$

که در آن  $i$ -نشان‌دهنده عامل مقابل است. با توجه به ماهیت بازی مجموع‌صفر، هر عامل تلاش می‌کند تا مطلوبیت خود را افزایش دهد، در حالی که مطلوبیت عامل دیگر به طور همزمان کاهش می‌یابد.

#### شبکه‌های هدف و بافر تجربه

مشابه DDPG، برای پایدار کردن آموزش، شبکه‌های هدف با میانگین‌گیری پولیاک به روزرسانی می‌شوند:

$$\phi_{i,\text{targ}} \leftarrow \rho\phi_{i,\text{targ}} + (1 - \rho)\phi_i$$

$$\theta_{i,\text{targ}} \leftarrow \rho\theta_{i,\text{targ}} + (1 - \rho)\theta_i$$

همچنین، از یک بافر تکرار بازی مشترک برای ذخیره تجربیات استفاده می‌شود که شامل وضعیت‌ها، اعمال و پاداش‌های همه عامل‌هاست.

### ۴-۳-۶ اکتشاف در MA-DDPG

اکتشاف در MA-DDPG مشابه DDPG است، اما برای هر عامل به طور جداگانه اعمال می‌شود. در طی آموزش، به اعمال هر عامل نویز اضافه می‌شود:

$$a_i = \text{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}}) \quad (11-6)$$

که در آن  $\epsilon_i$  نویز اضافه شده به عامل  $i$  است.

### ۵-۳-۶ شبکه MA-DDPG برای بازی‌های دو عاملی مجموع صفر

در این بخش، شبکه MA-DDPG پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم [۵](#) در محیط پایتون با استفاده از کتابخانه PyTorch [\[۶۰\]](#) پیاده‌سازی شده است.

---

## الگوریتم ۵ گرادیان سیاست عمیق قطعی چندعاملی برای بازی‌های مجموع صفر

---

- وروودی: پارامترهای اولیه سیاست عامل‌ها  $(\theta_1, \theta_2)$ , پارامترهای تابع  $Q(\phi_1, \phi_2)$ , بافر تکرار بازی خالی ( $\mathcal{D}$ )
- ۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید:  $\phi_{i,\text{targ}} \leftarrow \phi_i$ ,  $\theta_{i,\text{targ}} \leftarrow \theta_i$  برای  $i \in \{1, 2\}$
  - ۲: تا وقتی همگرایی رخ دهد:
  - ۳: مشاهدات  $(o_1, o_2)$  را دریافت کنید
  - ۴: برای هر عامل  $i$ , عمل  $a_i = \text{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}})$  را انتخاب کنید، به‌طوری که  $\epsilon_i \sim \mathcal{N}$  است
  - ۵: اعمال  $(a_1, a_2)$  را در محیط اجرا کنید
  - ۶: مشاهدات بعدی  $(o'_1, o'_2)$ , پاداش‌ها  $(r_1, r_2 = -r_1)$  و سیگنال پایان  $d$  را دریافت کنید
  - ۷: تجربه  $(o_1, o_2, a_1, a_2, r_1, r_2, o'_1, o'_2, d)$  را در بافر  $\mathcal{D}$  ذخیره کنید
  - ۸: اگر  $d = 1$  است، وضعیت محیط را بازنشانی کنید
  - ۹: اگر زمان بهروزرسانی فرا رسیده است:
  - ۱۰: به ازای هر تعداد بهروزرسانی:
  - ۱۱: یک دسته تصادفی از تجربیات،  $\{(o, a, r_1, r_2, o', d)\}$  از  $\mathcal{D}$  نمونه‌گیری کنید اهداف را محاسبه کنید:

$$y_1 = r_1 + \gamma(1 - d)Q_{\phi_1,\text{targ}}(o'_1, \mu_{\theta_1,\text{targ}}(o'_1), \mu_{\theta_2,\text{targ}}(o'_2))$$

$$y_2 = r_2 + \gamma(1 - d)Q_{\phi_2,\text{targ}}(o'_2, \mu_{\theta_2,\text{targ}}(o'_2), \mu_{\theta_1,\text{targ}}(o'_1))$$

تابع  $Q$  را با نزول گرادیان بهروزرسانی کنید: :۱۲

$$\nabla_{\phi_1} \frac{1}{|B|} \sum_{(o, a, r_1, r_2, o', d) \in B} (Q_{\phi_1}(o_1, a_1, a_2) - y_1)^2$$

$$\nabla_{\phi_2} \frac{1}{|B|} \sum_{(o, a, r_1, r_2, o', d) \in B} (Q_{\phi_2}(o_2, a_2, a_1) - y_2)^2$$

سیاست‌ها را با صعود گرادیان بهروزرسانی کنید: :۱۳

$$\nabla_{\theta_1} \frac{1}{|B|} \sum_{o \in B} Q_{\phi_1}(o_1, \mu_{\theta_1}(o_1), a_2)$$

$$\nabla_{\theta_2} \frac{1}{|B|} \sum_{o \in B} Q_{\phi_2}(o_2, \mu_{\theta_2}(o_2), a_1)$$

شبکه‌های هدف را بهروزرسانی کنید: :۱۴

$$\phi_{1,\text{targ}} \leftarrow \rho \phi_{1,\text{targ}} + (1 - \rho) \phi_1$$

$$\phi_{2,\text{targ}} \leftarrow \rho \phi_{2,\text{targ}} + (1 - \rho) \phi_2$$

$$\theta_{1,\text{targ}} \leftarrow \rho \theta_{1,\text{targ}} + (1 - \rho) \theta_1$$

$$\theta_{2,\text{targ}} \leftarrow \rho \theta_{2,\text{targ}} + (1 - \rho) \theta_2$$


---

## ۶-۳-۶ مزایای MA-DDPG در بازی‌های مجموع‌صفر

MA-DDPG چندین مزیت برای یادگیری در بازی‌های دوعلاملی مجموع‌صفر ارائه می‌دهد:

- **مقابله با غیرایستایی:** با استفاده از منتقدهایی که به اطلاعات کامل دسترسی دارند، مشکل غیرایستایی محیط از دید هر عامل حل می‌شود.
- **همگرایی بهتر:** در بازی‌های مجموع‌صفر، MA-DDPG معمولاً همگرایی بهتری نسبت به آموزش مستقل عامل‌ها با DDPG نشان می‌دهد.
- **یادگیری استراتژی‌های متقابل:** عامل‌ها می‌توانند استراتژی‌های متقابل پیچیده را یاد بگیرند که در آموزش مستقل امکان‌پذیر نیست.

در بازی‌های دوعلاملی مجموع‌صفر، این رویکرد به رقابت کامل بین عامل‌ها منجر می‌شود، که هر یک تلاش می‌کند بهترین استراتژی را در برابر استراتژی رقیب پیدا کند.

## ۴-۶ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه چندعلاملی

عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه چندعلاملی<sup>۱۰</sup> توسعه‌ای از الگوریتم TD3 برای محیط‌های چندعلاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های چندعلاملی مجموع‌صفر می‌پردازیم که در آن ترکیب ویژگی‌های TD3 با رویکرد چندعلاملی MADDPG به پایداری و کارایی بیشتر در یادگیری منجر می‌شود.

## ۱-۴-۶ چالش‌های یادگیری تقویتی در محیط‌های چندعلاملی و راه حل MATD3

در محیط‌های چندعلاملی، عامل‌ها همزمان سیاست‌های خود را تغییر می‌دهند که باعث غیرایستایی محیط از دید هر عامل می‌شود. علاوه بر این، بیش برآورد تابع  $Q$  که در DDPG دیده می‌شود، در محیط‌های چندعلاملی می‌تواند تشدید شود.

MATD3 هر دو چالش را با ترکیب رویکردهای زیر حل می‌کند:

- **آموزش مرکزی، اجرای غیرمرکز:** مشابه MADDPG، از منتقدهایی استفاده می‌کند که به اطلاعات کامل دسترسی دارند.

<sup>10</sup>Multi-Agent Twin Delayed Deep Deterministic Policy Gradient (MATD3)

- منتقدهای دوگانه: برای هر عامل، از دو شبکه منتقد استفاده می‌کند تا بیش‌برآورد تابع  $Q$  را کاهش دهد.

- بهروزرسانی‌های تاخیری سیاست: سیاست‌ها را با تواتر کمتری نسبت به منتقدها بهروزرسانی می‌کند.

## ۲-۴-۶ معماری MATD3 در بازی‌های مجموع‌صفر

در یک بازی چندعاملی مجموع‌صفر، هر عامل دارای شبکه‌های زیر است:

- شبکه بازیگر:  $\mu_{\theta_i}(o_i)$  که مشاهدات محلی  $o_i$  را به اعمال  $a_i$  نگاشت می‌کند.
  - شبکه‌های منتقد دوگانه:  $Q_{\phi_{i,2}}(o_i, a_1, a_2)$  و  $Q_{\phi_{i,1}}(o_i, a_1, a_2)$  که ارزش حالت-عمل را تخمین می‌زنند.
  - شبکه‌های هدف: برای پایدارسازی آموزش، از نسخه‌های هدف بازیگر و منتقدها استفاده می‌شود.
- در بازی‌های مجموع‌صفر، پاداش‌ها رابطه  $r_1 + r_2 = 0$  دارند، بنابراین  $r_2 = -r_1$  است.

## ۳-۴-۶ آموزش MATD3

فرایند آموزش MATD3 به شرح زیر است:

یادگیری تابع  $Q$

برای هر عامل  $i \in \{1, 2\}$  و هر منتقد  $j \in \{1, 2\}$ ، تابع  $Q$  با کمینه کردن خطای میانگین مربعات بلمن بهروزرسانی می‌شود:

$$L(\phi_{i,j}, \mathcal{D}) = \underset{(o, a, r_i, o', d) \sim \mathcal{D}}{\text{E}} \left[ \left( Q_{\phi_{i,j}}(o_i, a_1, a_2) - y_i \right)^2 \right] \quad (12-6)$$

که در آن  $y_i$  هدف برای عامل  $i$  است:

$$y_i = r_i + \gamma(1-d) \min_{j=1,2} Q_{\phi_{i,j,\text{targ}}}(o'_i, \mu_{\theta_{1,\text{targ}}}(o'_1), \mu_{\theta_{2,\text{targ}}}(o'_2)) \quad (13-6)$$

استفاده از عملگر حداقل روی دو منتقد، بیش برآورد را کاهش می‌دهد که منجر به تخمین‌های محتاطانه‌تر و پایدارتر می‌شود.

### یادگیری سیاست با تاخیر

سیاست هر عامل با تاخیر (معمولًاً پس از هر دو بهروزرسانی منتقدها) و با بیشینه کردن تابع  $Q$  اول بهروزرسانی می‌شود:

$$\max_{\theta_i} \mathbb{E}_{o \sim \mathcal{D}} [Q_{\phi_{i,1}}(o_i, \mu_{\theta_i}(o_i), \mu_{\theta_{-i}}(o_{-i}))] \quad (14-6)$$

بهروزرسانی تاخیری سیاست اجازه می‌دهد تا منتقدها قبل از تغییر سیاست به مقادیر دقیق‌تری همگرا شوند.

### شبکه‌های هدف

مشابه TD3، شبکه‌های هدف با میانگین‌گیری پولیاک بهروزرسانی می‌شوند:

$$\begin{aligned} \phi_{i,j,\text{targ}} &\leftarrow \rho \phi_{i,j,\text{targ}} + (1 - \rho) \phi_{i,j} \quad \text{برای } j = 1, 2 \\ \theta_{i,\text{targ}} &\leftarrow \rho \theta_{i,\text{targ}} + (1 - \rho) \theta_i \end{aligned}$$

### ۴-۶-۶ اکتشاف در MATD3

اکتشاف در MATD3 با افروzen نویز به اعمال هر عامل انجام می‌شود:

$$a_i = \text{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}}) \quad (15-6)$$

که در آن  $\epsilon_i \sim \mathcal{N}(0, \sigma_i)$  است و مقدار  $\sigma_i$  به مرور زمان کاهش می‌یابد.

## ۵-۴-۶ شبکه برای بازی‌های چندعاملی مجموع صفر MATD3

در این بخش، شبکه الگوریتم MATD3 پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم [۶](#) در محیط پایتون با استفاده از کتابخانه PyTorch [\[۶۰\]](#) پیاده‌سازی شده است.

## الگوریتم ۶ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه دوعلاملی

وروودی: پارامترهای اولیه سیاست عاملها  $(\phi_{1,1}, \phi_{1,2}, \phi_{2,1}, \phi_{2,2})$ ، پارامترهای توابع  $Q$ ، بافر تکرار بازی خالی  $(\mathcal{D})$

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید:

$$j \in \{1, 2\} \text{ برای } i \in \{1, 2\} \text{ و } \phi_{i,j,\text{targ}} \leftarrow \phi_{i,j}, \theta_{i,\text{targ}} \leftarrow \theta_i$$

۲: تا وقتی همگرایی رخ دهد:

۳: مشاهدات  $(o_1, o_2)$  را دریافت کنید

۴: برای هر عامل  $i$ ، عمل  $a_i = \text{clip}(\mu_{\theta_i}(o_i) + \epsilon_i, a_{\text{Low}}, a_{\text{High}})$  را انتخاب کنید، بهطوری که  $\sim \mathcal{N}(0, \sigma_i)$  است

۵: اعمال  $(a_1, a_2)$  را در محیط اجرا کنید

۶: مشاهدات بعدی  $(o'_1, o'_2)$ ، پاداشها  $(r_1, r_2 = -r_1)$  و سیگنال پایان  $d$  را دریافت کنید

۷: تجربه  $(o_1, o_2, a_1, a_2, r_1, r_2, o'_1, o'_2, d)$  را در بافر  $\mathcal{D}$  ذخیره کنید

۸: اگر  $d = 1$  است، وضعیت محیط را بازنشانی کنید

۹: اگر زمان بهروزرسانی فرا رسیده است:

۱۰: به ازای  $j$  در هر تعداد بهروزرسانی:

۱۱: یک دسته تصادفی از تجربیات،  $\{(o, a, r_1, r_2, o', d)\}$  از  $\mathcal{D}$  نمونهگیری کنید

۱۲: اهداف را محاسبه کنید:

$$y_1 = r_1 + \gamma(1-d) \min_{k=1,2} Q_{\phi_{1,k},\text{targ}}(o'_1, \mu_{\theta_{1,\text{targ}}}(o'_1), \mu_{\theta_{2,\text{targ}}}(o'_2))$$

$$y_2 = r_2 + \gamma(1-d) \min_{k=1,2} Q_{\phi_{2,k},\text{targ}}(o'_2, \mu_{\theta_{2,\text{targ}}}(o'_2), \mu_{\theta_{1,\text{targ}}}(o'_1))$$

۱۳: توابع  $Q$  را با نزول گرادیان بهروزرسانی کنید:

$$\nabla_{\phi_{1,k}} \frac{1}{|B|} \sum_B (Q_{\phi_{1,k}}(o_1, a_1, a_2) - y_1)^2 \quad \text{برای } k = 1, 2$$

$$\nabla_{\phi_{2,k}} \frac{1}{|B|} \sum_B (Q_{\phi_{2,k}}(o_2, a_2, a_1) - y_2)^2 \quad \text{برای } k = 1, 2$$

۱۴: اگر باقیمانده  $j$  بر تاخیر سیاست برابر ۰ باشد:

۱۵: سیاستها را با صعود گرادیان بهروزرسانی کنید:

$$\nabla_{\theta_1} \frac{1}{|B|} \sum_{o \in B} Q_{\phi_{1,1}}(o_1, \mu_{\theta_1}(o_1), a_2)$$

$$\nabla_{\theta_2} \frac{1}{|B|} \sum_{o \in B} Q_{\phi_{2,1}}(o_2, \mu_{\theta_2}(o_2), a_1)$$

۱۶: شبکه‌های هدف را بهروزرسانی کنید:

$$\phi_{i,k,\text{targ}} \leftarrow \rho \phi_{i,k,\text{targ}} + (1-\rho) \phi_{i,k} \quad \text{برای } i, k \in \{1, 2\}$$

$$\theta_{i,\text{targ}} \leftarrow \rho \theta_{i,\text{targ}} + (1-\rho) \theta_i \quad \text{برای } i \in \{1, 2\}$$

## ۶-۴-۶ مزایای MATD3 در بازی‌های مجموع‌صفر

MATD3 مزایای زیر را نسبت به MADDPG در بازی‌های چندعاملی مجموع‌صفر ارائه می‌دهد:

- پایداری بیشتر: با استفاده از منتقدهای دوگانه، بیش‌برآورد تابع  $Q$  که در محیط‌های غیرایستای چند-عاملی شدیدتر است، کاهش می‌یابد.
- یادگیری کارآمدتر: به روزسانی‌های تاخیری سیاست اجازه می‌دهد منتقدها به تخمین‌های دقیق‌تری دست یابند، که منجر به بهبود کیفیت یادگیری سیاست می‌شود.
- مقاومت در برابر نویز: ترکیب منتقدهای دوگانه با رویکرد آموزش متمنکز، مقاومت الگوریتم در برابر نویز و تغییرات محیط را افزایش می‌دهد.
- همگرایی بهتر: بهبودهای TD3 در کنار رویکرد چندعاملی، به همگرایی سریع‌تر و پایدارتر در بازی‌های رقابتی منجر می‌شود.

در مجموع، MATD3 ترکیبی از بهترین ویژگی‌های TD3 و MADDPG را ارائه می‌دهد که آن را به گزینه‌ای مناسب برای یادگیری سیاست‌های پیچیده در بازی‌های چندعاملی مجموع‌صفر تبدیل می‌کند.

## ۵-۶ عامل عملگر نقاد نرم چندعاملی

عامل عملگر نقاد نرم دواعمالی<sup>۱۱</sup> توسعه‌ای از الگوریتم SAC برای محیط‌های چندعاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های چندعاملی مجموع‌صفر می‌پردازیم که در آن ترکیب ویژگی‌های SAC با رویکرد چندعاملی به پایداری و کارایی بیشتر در یادگیری منجر می‌شود.

## ۱-۵-۶ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه حل MASAC

در محیط‌های چندعاملی، عامل‌ها همزمان سیاست‌های خود را تغییر می‌دهند که باعث غیرایستایی محیط از دید هر عامل می‌شود. علاوه بر این، چالش‌های مربوط به تعادل اکتشاف-بهره‌برداری در محیط‌های چندعاملی پیچیده‌تر است.

MASAC این چالش‌ها را با ترکیب رویکردهای زیر حل می‌کند:

<sup>11</sup> Multi-Agent Soft Actor-Critic (MASAC)

- آموزش متمرکز، اجرای غیر متمرکز: مشابه MADDPG، از منتقدهایی استفاده می‌کند که به اطلاعات کامل دسترسی دارند.
- سیاست‌های تصادفی: برخلاف MADDPG و MATD3 که سیاست‌های قطعی دارند، MASAC از سیاست‌های تصادفی استفاده می‌کند.
- تنظیم آنتروپی: با استفاده از تنظیم آنتروپی، اکتشاف و همگرایی به سیاست‌های بهتر را بهبود می‌بخشد.
- منتقدهای دوگانه: برای هر عامل، از دو شبکه منتقد استفاده می‌کند تا بیش برآورد تابع  $Q$  را کاهش دهد.

## ۲-۵-۶ معماری MASAC در بازی‌های مجموع صفر

- در یک بازی چندعاملی مجموع صفر، هر عامل دارای شبکه‌های زیر است:
- شبکه بازیگر:  $\pi_{\theta_i}(a_i|o_i)$  که توزیع احتمال اعمال را با توجه به مشاهدات محلی تعیین می‌کند.
  - شبکه‌های منتقد دوگانه:  $Q_{\phi_{i,2}}(o_i, a_1, a_2)$  و  $Q_{\phi_{i,1}}(o_i, a_1, a_2)$  که ارزش حالت-عمل را تخمین می‌زنند.
  - شبکه‌های هدف: برای پایدارسازی آموزش، از نسخه‌های هدف منتقدها استفاده می‌شود.
- در بازی‌های مجموع صفر، پاداش‌ها رابطه  $r_1 + r_2 = -r_1 - r_2$  دارند، بنابراین  $r_1 = r_2 = 0$  است.

## ۳-۵-۶ آموزش MASAC

فرایند آموزش MASAC به شرح زیر است:

یادگیری تابع  $Q$

برای هر عامل  $i \in \{1, 2\}$  و هر منتقد  $j \in \{1, 2\}$ ، تابع  $Q$  با کمینه کردن خطای میانگین مربعات بلمن بهروزرسانی می‌شود:

$$L(\phi_{i,j}, \mathcal{D}) = \underset{(o,a,r_i,o',d) \sim \mathcal{D}}{\text{E}} \left[ \left( Q_{\phi_{i,j}}(o_i, a_1, a_2) - y_i \right)^2 \right] \quad (16-6)$$

که در آن  $y_i$  هدف برای عامل  $i$  است:

$$y_i = r_i + \gamma(1-d) \left( \min_{j=1,2} Q_{\phi_{i,j},\text{targ}}(o'_i, \tilde{a}'_1, \tilde{a}'_2) - \alpha_i \log \pi_{\theta_i}(\tilde{a}'_i | o'_i) \right) \quad (17-6)$$

که در آن  $(\cdot | o'_i) \sim \pi_{\theta_i}$  است. استفاده از عملگر حداقل روی دو منتقد، بیش برآورد را کاهش می‌دهد که منجر به تخمین‌های محتاطانه‌تر و پایدارتر می‌شود.

### یادگیری سیاست

سیاست هر عامل با بیشینه کردن ترکیبی ازتابع  $Q$  و آنتروپی به روزرسانی می‌شود:

$$\max_{\theta_i} \mathbb{E}_{o \sim \mathcal{D}} \left[ \min_{j=1,2} Q_{\phi_{i,j}}(o_i, \tilde{a}_i, a_{-i}) - \alpha_i \log \pi_{\theta_i}(\tilde{a}_i | o_i) \right] \quad (18-6)$$

که در آن  $\tilde{a}_i \sim \pi_{\theta_i}(\cdot | o_i)$  است و از ترفندهای پارامترسازی مجدد برای استخراج گرادیان استفاده می‌شود:

$$\tilde{a}_{i,\theta_i}(o_i, \xi_i) = \tanh(\mu_{\theta_i}(o_i) + \sigma_{\theta_i}(o_i) \odot \xi_i), \quad \xi_i \sim \mathcal{N} \quad (19-6)$$

### شبکه‌های هدف

مشابه SAC، شبکه‌های هدف منتقد با میانگین‌گیری پولیاک به روزرسانی می‌شوند:

$$\phi_{i,j,\text{targ}} \leftarrow \rho \phi_{i,j,\text{targ}} + (1-\rho) \phi_{i,j} \quad \text{برای } j = 1, 2 \quad (20-6)$$

### تنظیم ضریب آنتروپی

یکی از مزایای MASAC، توانایی تنظیم خودکار ضریب آنتروپی  $\alpha_i$  برای هر عامل است که می‌تواند با استفاده از یک تابع هزینه مجزا بهینه شود:

$$\min_{\alpha_i} \mathbb{E}_{o \sim \mathcal{D}, \tilde{a}_i \sim \pi_{\theta_i}} \left[ -\alpha_i \left( \log \pi_{\theta_i}(\tilde{a}_i | o_i) + H_{\text{target}} \right) \right] \quad (21-6)$$

که در آن  $H_{\text{target}}$  آنتروپی هدف است که به عنوان یک ابرپارامتر تعیین می‌شود.

#### ۴-۵-۶ اکتشاف در MASAC

اکتشاف در MASAC به صورت ذاتی از طریق سیاست‌های تصادفی و تنظیم آنتروپی انجام می‌شود. برخلاف MADDPG و MATD3 که به افزودن نویز به اعمال نیاز دارند، MASAC اعمال را مستقیماً از توزیع احتمال سیاست نمونه‌گیری می‌کند:

$$a_i \sim \pi_{\theta_i}(\cdot | O_i) \quad (22-6)$$

این رویکرد امکان اکتشاف ساختاریافته‌تر و کارآمدتر را فراهم می‌کند که در محیط‌های چندعاملی پیچیده مفید است.

#### ۵-۵-۶ شبکه MASAC برای بازی‌های چندعاملی مجموع صفر

در این بخش، شبکه الگوریتم MASAC پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم [۷](#) در محیط پایتون با استفاده از کتابخانه PyTorch [\[۶۰\]](#) پیاده‌سازی شده است.

## الگوریتم ۷ عامل عملگر نقاد نرم دو عاملی

وروودی: پارامترهای اولیه سیاست عاملها  $(\theta_1, \theta_2)$ ، پارامترهای توابع  $Q(\phi_{1,1}, \phi_{1,2}, \phi_{2,1}, \phi_{2,2})$ ، ضرایب آنتروپی  $(\alpha_1, \alpha_2)$ ، بافر تکرار بازی خالی  $(\mathcal{D})$

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید:

$$j \in \{1, 2\} \text{ و } i \in \{1, 2\} \text{ برای } \phi_{i,j,\text{targ}} \leftarrow \phi_{i,j}$$

۲: تا وقتی همگرایی رخ دهد:

۳: مشاهدات  $(o_1, o_2)$  را دریافت کنید

۴: برای هر عامل  $i$ ، عمل  $a_i \sim \pi_{\theta_i}(\cdot | o_i)$  را انتخاب کنید

۵: اعمال  $(a_1, a_2)$  را در محیط اجرا کنید

۶: مشاهدات بعدی  $(o'_1, o'_2)$ ، پاداشها  $(r_1, r_2 = -r_1)$  و سیگنال پایان  $d$  را دریافت کنید

۷: تجربه  $\mathcal{D}$  را در بافر ذخیره کنید

۸: اگر  $d = 1$  است، وضعیت محیط را بازنشانی کنید

۹: اگر زمان بهروزرسانی فرا رسیده است:

۱۰: به ازای هر تعداد بهروزرسانی:

۱۱: یک دسته تصادفی از تجربیات،  $\{(o, a, r_1, r_2, o', d)\}$  از  $\mathcal{D}$  نمونه‌گیری کنید

۱۲: اهداف را محاسبه کنید:

$$y_1 = r_1 + \gamma(1-d) \left( \min_{j=1,2} Q_{\phi_{1,j,\text{targ}}}(o'_1, \tilde{a}'_1, \tilde{a}'_2) - \alpha_1 \log \pi_{\theta_1}(\tilde{a}'_1 | o'_1) \right)$$

$$y_2 = r_2 + \gamma(1-d) \left( \min_{j=1,2} Q_{\phi_{2,j,\text{targ}}}(o'_2, \tilde{a}'_2, \tilde{a}'_1) - \alpha_2 \log \pi_{\theta_2}(\tilde{a}'_2 | o'_2) \right)$$

۱۳: توابع  $Q$  را با نزول گرادیان بهروزرسانی کنید:

$$\nabla_{\phi_{1,j}} \frac{1}{|B|} \sum_B (Q_{\phi_{1,j}}(o_1, a_1, a_2) - y_1)^2 \quad \text{برای } j = 1, 2$$

$$\nabla_{\phi_{2,j}} \frac{1}{|B|} \sum_B (Q_{\phi_{2,j}}(o_2, a_2, a_1) - y_2)^2 \quad \text{برای } j = 1, 2$$

۱۴: سیاست‌ها را با صعود گرادیان بهروزرسانی کنید:

$$\nabla_{\theta_1} \frac{1}{|B|} \sum_{o \in B} \left[ \min_{j=1,2} Q_{\phi_{1,j}}(o_1, \tilde{a}_{1,\theta_1}(o_1, \xi_1), a_2) - \alpha_1 \log \pi_{\theta_1}(\tilde{a}_{1,\theta_1}(o_1, \xi_1) | o_1) \right]$$

$$\nabla_{\theta_2} \frac{1}{|B|} \sum_{o \in B} \left[ \min_{j=1,2} Q_{\phi_{2,j}}(o_2, \tilde{a}_{2,\theta_2}(o_2, \xi_2), a_1) - \alpha_2 \log \pi_{\theta_2}(\tilde{a}_{2,\theta_2}(o_2, \xi_2) | o_2) \right]$$

۱۵: ضرایب آنتروپی را با نزول گرادیان بهروزرسانی کنید (اختیاری):

$$\nabla_{\alpha_1} \frac{1}{|B|} \sum_{o \in B} -\alpha_1 \left( \log \pi_{\theta_1}(\tilde{a}_{1,\theta_1}(o_1, \xi_1) | o_1) + H_{\text{target}} \right)$$

$$\nabla_{\alpha_2} \frac{1}{|B|} \sum_{o \in B} -\alpha_2 \left( \log \pi_{\theta_2}(\tilde{a}_{2,\theta_2}(o_2, \xi_2) | o_2) + H_{\text{target}} \right)$$

۱۶: شبکه‌های هدف را بهروزرسانی کنید:

$$\phi_{i,j,\text{targ}} \leftarrow \rho \phi_{i,j,\text{targ}} + (1-\rho) \phi_{i,j} \quad \text{برای } i, j \in \{1, 2\}$$

## ۶-۵-۶ مزایای MASAC در بازی‌های مجموع‌صفر

MASAC مزایای زیر را نسبت به سایر الگوریتم‌های چندعاملی در بازی‌های چندعاملی مجموع‌صفر ارائه می‌دهد:

- **اکتشاف بهتر:** استفاده از سیاست‌های تصادفی و تنظیم آنتروپی، اکتشاف فضای حالت-عمل را بهبود می‌بخشد که برای یافتن راه حل‌های بهینه در بازی‌های دو عاملی ضروری است.
- **ثبات بیشتر:** ترکیب منتقدهای دوگانه با تنظیم آنتروپی، یادگیری را پایدارتر می‌کند و از همگرایی زودهنگام به سیاست‌های ضعیف جلوگیری می‌کند.
- **سازگاری با محیط‌های پیچیده:** توانایی تنظیم خودکار تعادل بین اکتشاف و بهره‌برداری، MASAC را برای محیط‌های چندعاملی پیچیده مناسب می‌سازد.
- **عملکرد بهتر در مسائل با چندین بهینه محلی:** سیاست‌های تصادفی می‌توانند از دام‌های بهینه محلی فرار کنند و به راه حل‌های بهتر برسند.

در مجموع، MASAC ترکیبی از ویژگی‌های مثبت SAC و رویکردهای چندعاملی را ارائه می‌دهد که آن را به گزینه‌ای قدرتمند برای یادگیری سیاست‌های پیچیده در بازی‌های چندعاملی مجموع‌صفر تبدیل می‌کند، به ویژه در محیط‌هایی که اکتشاف کارآمد و سیاست‌های تصادفی اهمیت دارند.

## ۶-۶ عامل بهینه‌سازی سیاست مجاور چندعاملی

عامل بهینه‌سازی سیاست مجاور دو عاملی<sup>۱۲</sup> توسعه‌ای از الگوریتم PPO برای محیط‌های چندعاملی است. در این بخش، به بررسی این الگوریتم در چارچوب بازی‌های چندعاملی مجموع‌صفر می‌پردازیم که در آن ترکیب ویژگی‌های PPO با رویکرد چندعاملی به پایداری و کارایی بیشتر در یادگیری منجر می‌شود.

## ۱-۶-۶ چالش‌های یادگیری تقویتی در محیط‌های چندعاملی و راه حل MAPPO

در محیط‌های چندعاملی، عامل‌ها هم‌زمان سیاست‌های خود را تغییر می‌دهند که باعث غیرایستایی محیط از دید هر عامل می‌شود. این چالش با پیچیدگی‌های ذاتی الگوریتم‌های مبتنی بر گرادیان سیاست مانند PPO ترکیب می‌شود.

<sup>12</sup>Multi-Agent Proximal Policy Optimization (MAPPO)

MAPPO این چالش‌ها را با ترکیب رویکردهای زیر حل می‌کند:

- آموزش مرکزی، اجرای غیرمرکز: مشابه سایر الگوریتم‌های چندعاملی، از منتقدهایی استفاده می‌کند که به اطلاعات کامل دسترسی دارند، اما بازیگران تنها به مشاهدات محلی خود دسترسی دارند.
- بهروزرسانی کلیپ‌شده: استفاده از مکانیسم کلیپ شده PPO برای محدود کردن بهروزرسانی‌های سیاست، که به پایداری بیشتر در یادگیری چندعاملی کمک می‌کند.
- بافر تجربه مشترک: استفاده از یک بافر تجربه مشترک که تعاملات بین عامل‌ها را ثبت می‌کند.

## ۲-۶-۶ معماری MAPPO در بازی‌های مجموع صفر

در یک بازی چندعاملی مجموع صفر، هر عامل دارای شبکه‌های زیر است:

- شبکه بازیگر:  $\pi_{\theta_i}(a_i|o_i)$  که توزیع احتمال اعمال را با توجه به مشاهدات محلی تعیین می‌کند.
- شبکه منتقد:  $V_{\phi_i}(o_i, a_1, a_2)$  که ارزش حالت را تخمین می‌زند و برای محاسبه تابع مزیت استفاده می‌شود.

در بازی‌های مجموع صفر، پاداش‌ها رابطه  $r_1 + r_2 = 0$  دارند، بنابراین  $r_2 = -r_1$  است.

## ۳-۶-۶ آموزش MAPPO

فرایند آموزش MAPPO به شرح زیر است:

### جمع‌آوری تجربیات

در هر تکرار، عامل‌ها با استفاده از سیاست‌های فعلی خود در محیط تعامل می‌کنند و مجموعه‌ای از مسیرها را جمع‌آوری می‌کنند:

$$\mathcal{D}_k = \{(o_1^t, o_2^t, a_1^t, a_2^t, r_1^t, r_2^t, o_1^{t+1}, o_2^{t+1})\} \quad (23-6)$$

## محاسبه مزیت

برای هر عامل  $i \in \{1, 2\}$ ، تابع مزیت با استفاده از تابع ارزش فعلی محاسبه می‌شود. روش‌های مختلفی برای محاسبه مزیت وجود دارد؛ یک روش متداول استفاده از تخمین‌زننده مزیت تعمیم‌پافته (GAE) است:

$$\hat{A}_i^t = \sum_{l=0}^{\infty} (\gamma \lambda)^l \delta_{i,t+l} \quad (24-6)$$

که در آن  $\delta_{i,t} = r_i^t + \gamma V_{\phi_i}(o_i^{t+1}) - V_{\phi_i}(o_i^t)$  است.

## بهروزرسانی سیاست

سیاست هر عامل با بیشینه کردن تابع هدف PPO-Clip بهروزرسانی می‌شود:

$$\max_{\theta_i} \mathbb{E}_{(o_i, a_i) \sim \mathcal{D}_k} \left[ \min \left( \frac{\pi_{\theta_i}(a_i | o_i)}{\pi_{\theta_{i,k}}(a_i | o_i)} \hat{A}_i, \text{clip} \left( \frac{\pi_{\theta_i}(a_i | o_i)}{\pi_{\theta_{i,k}}(a_i | o_i)}, 1 - \epsilon, 1 + \epsilon \right) \hat{A}_i \right) \right] \quad (25-6)$$

یا با استفاده از همان فرمول‌بندی ساده‌تر:

$$\max_{\theta_i} \mathbb{E}_{(o_i, a_i) \sim \mathcal{D}_k} \left[ \min \left( \frac{\pi_{\theta_i}(a_i | o_i)}{\pi_{\theta_{i,k}}(a_i | o_i)} \hat{A}_i, g(\epsilon, \hat{A}_i) \right) \right] \quad (26-6)$$

که تابع  $g$  به صورت زیر تعریف شده‌است:

$$g(\epsilon, A) = \begin{cases} (1 + \epsilon)A & A \geq 0 \\ (1 - \epsilon)A & A < 0 \end{cases} \quad (27-6)$$

## بهروزرسانی منتقد

تابع ارزش هر عامل با کمینه کردن خطای میانگین مربعات بهروزرسانی می‌شود:

$$\min_{\phi_i} \mathbb{E}_{(o_i, \hat{R}_i) \sim \mathcal{D}_k} \left[ \left( V_{\phi_i}(o_i) - \hat{R}_i \right)^2 \right] \quad (28-6)$$

که در آن  $\hat{R}_i$  بازده تنزیل شده برای عامل  $i$  است.

## ۴-۶-۶ اکتشاف در MAPPO

اکتشاف در MAPPO به صورت ذاتی از طریق سیاست‌های تصادفی انجام می‌شود. برخلاف الگوریتم‌های مبتنی بر DDPG که به افزودن نویز به اعمال نیاز دارند، MAPPO از توزیع احتمال سیاست برای اکتشاف استفاده می‌کند:

$$a_i \sim \pi_{\theta_i}(\cdot | O_i) \quad (29-6)$$

این رویکرد اکتشاف سیاست‌محور، در ترکیب با مکانیسم کلیپ PPO که از بهروزرسانی‌های بزرگ سیاست جلوگیری می‌کند، به ثبات بیشتر در یادگیری چندعاملی کمک می‌کند.

## ۵-۶-۶ شبکه‌کد MAPPO برای بازی‌های چندعاملی مجموع‌صفر

در این بخش، شبکه‌کد الگوریتم MAPPO پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم [۸](#) در محیط پایتون با استفاده از کتابخانه PyTorch [\[۶۰\]](#) پیاده‌سازی شده است.

---

## الگوریتم ۸ عامل بهینه‌سازی سیاست مجاور دوعلایی

---

وروودی: پارامترهای اولیه سیاست عاملها  $(\theta_1, \theta_2)$ ، پارامترهای تابع ارزش  $(\phi_1, \phi_2)$

: ۱ به ازای ...  $k = 0, 1, 2, \dots$

: ۲ مجموعه‌ای از مسیرها به نام  $\mathcal{D}_k = \{(o_1^t, o_2^t, a_1^t, a_2^t, r_1^t, r_2^t, o_1^{t+1}, o_2^{t+1})\}$  با اجرای سیاست‌های  $\pi_{\theta_1}$  و  $\pi_{\theta_2}$  در محیط جمع‌آوری شود.

: ۳ برای هر عامل  $i$ ، پاداش‌های باقی‌مانده  $\hat{R}_i^t$  را محاسبه کنید.

: ۴ برای هر عامل  $i$ ، برآوردهای مزیت  $\hat{A}_i^t$  را با استفاده از تابع ارزش فعلی  $V_{\phi_i}$  محاسبه کنید.

: ۵ برای هر عامل  $i$ ، سیاست را با به حداقل رساندن تابع هدف PPO-Clip بهروزرسانی کنید:

$$\theta_{i,k+1} = \arg \max_{\theta_i} \frac{1}{|\mathcal{D}_k|} \sum_{(o_i, a_i) \in \mathcal{D}_k} \min \left( \frac{\pi_{\theta_i}(a_i | o_i)}{\pi_{\theta_{i,k}}(a_i | o_i)} \hat{A}_i, g(\epsilon, \hat{A}_i) \right)$$

: ۶ برای هر عامل  $i$ ، تابع ارزش را با رگرسیون بر روی میانگین مربعات خطأ بهروزرسانی کنید:

$$\phi_{i,k+1} = \arg \min_{\phi_i} \frac{1}{|\mathcal{D}_k|} \sum_{(o_i) \in \mathcal{D}_k} \left( V_{\phi_i}(o_i) - \hat{R}_i \right)^2$$

---

## ۶-۶-۶ مزایای MAPPO در بازی‌های مجموع صفر

MAPPO مزایای زیر را نسبت به سایر الگوریتم‌های چندعاملی مجموع صفر ارائه می‌دهد:

- پایداری یادگیری: مکانیسم کلیپ PPO از بهروزرسانی‌های بزرگ سیاست جلوگیری می‌کند که به پایداری بیشتر در محیط‌های غیرایستای چندعاملی منجر می‌شود.

- کارایی نمونه: نسبت به الگوریتم‌های خارج از سیاست مانند MATD3 و MASAC، MAPPO معمولاً کارایی نمونه بهتری دارد و به داده‌های کمتری برای یادگیری نیاز دارد.

- اکتشاف سیاست‌محور: اکتشاف ذاتی از طریق سیاست‌های تصادفی به جای افزودن نویز به اعمال، به اکتشاف کارآمدتر فضای حالت-عمل کمک می‌کند.

- مقیاس‌پذیری: MAPPO به راحتی به سیستم‌های با تعداد بیشتری از عامل‌ها قابل گسترش است، اگرچه در این پژوهش بر بازی‌های دوعلایی تمرکز شده است.

در مجموع، MAPPO ترکیبی از سادگی و کارایی PPO با رویکردهای چندعاملی را ارائه می‌دهد که آن را به گزینه‌ای قدرتمند برای یادگیری در بازی‌های چندعاملی مجموع صفر تبدیل می‌کند.

## فصل ۷

# ارزیابی عملکرد سخت افزار در حلقه عامل

ارزیابی عملکرد عامل یادگیری تقویتی صرفاً در حوزه‌ی شبیه‌سازی رایانه‌ای، گرچه در مراحل اولیه‌ی طراحی روشی کم‌هزینه و سریع است، نمی‌تواند تمام محدودیت‌های سخت افزاری سامانه‌ی پروازی را بازنمایی کند. در شبیه‌سازی‌های ایده‌آل، منابع پردازشی عمل‌آن محدود فرض می‌شود و تأخیر میان اجزای نرم افزار یا نویز ناشی از واسطه‌ای الکترونیکی به حداقل می‌رسد. چنین ساده‌سازی‌هایی ممکن است منجر به خوبی بیش از حد در برآورده میزان پایداری، دقت ریدیابی و الزامات توان مصرفی الگوریتم کنترل شود.

روش سخت افزار در حلقه<sup>۱</sup> راهکاری پذیرفته شده برای پل زدن میان شبیه‌سازی و فضای واقعی است. در این چارچوب، دو زیر سامانه‌ی مجزا اما پیوسته تعریف می‌شود:

- لایه‌ی میزبان (Host) که دینامیک مدار، مدل نیروهای اغتشاشی و حسگرهای مجازی را در بسامد ۱۰۰ Hz با استفاده از ROS شبیه‌سازی می‌کند.

- لایه‌ی هدف (Target) که یک رایانه‌ی تعبیه شده کم‌صرف (در این پروژه، Raspberry Pi 3 B) را نمایندگی می‌کند و چرخه‌ی ادراک-تصمیم-اقدام شامل تخمین حالت، استنتاج شبکه‌ی عصبی و تولید فرمان پیشران را در زمان واقعی اجرا می‌کند.

پیاده‌سازی HIL سه مزیت کلیدی دارد: (۱) امکان اندازه‌گیری دقیق تأخیر انتها-به-انتها و Jitter ناشی از محدودیت‌های پردازشی و (۲) مشاهده‌ی اثر نویز واقعی مبدل‌های ADC/DAC و افت ولتاژ منبع تغذیه بر پایداری کنترل. نتایج به دست آمده از چنین بستری اطمینان می‌دهد که سیاست RL، پس از فشرده‌سازی INT8 Quantization و اجرا بر سخت افزار کلاس پرواز، همچنان معیارهای عملکرد تعریف شده خود را برآورده می‌کند.

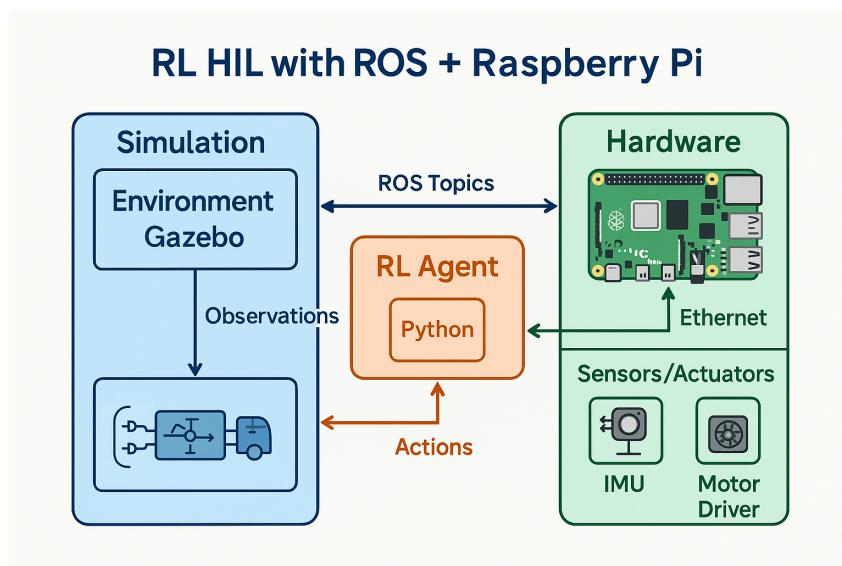
<sup>1</sup>Hardware-in-the-Loop (HIL).

اهداف فصل حاضر را می‌توان به صورت زیر خلاصه کرد:

۱. تشریح معماری آزمایشگاهی HIL و توجیه انتخاب اجزای سخت‌افزاری و نرم‌افزاری.
  ۲. تبیین رویه‌ی یکپارچه‌سازی گره‌های ROS میان میزبان و هدف با تضمین بسامد ۱۰۰ Hz.
  ۳. کمی‌سازی تأخیر و توان در چرخه‌ی کنترل و تحلیل اثر آن بر اینمنی مأموریت.
  ۴. ارائه‌ی سناریوهای اعتبارسنجی و نتایج مقدماتی جهت مقایسه با شبیه‌سازی صرف.
- ساختمان فصل به ترتیب در بخش‌های ۱-۶ تا ۱-۷ به جزئیات هر یک از اهداف فوق می‌پردازد.

## ۱-۷ پیکربندی آزمایشگاهی

به منظور ارزیابی زمان واقعی<sup>۲</sup> سیاست یادگیری تقویتی، یک بستر سخت‌افزار در حلقه با معماری دولایه طراحی شد که رایانه‌ی میزبان وظیفه‌ی شبیه‌سازی دینامیک و حسگرها را بر عهده دارد و رایانه‌ی هدف چرخه‌ی ادراک-تصمیم-اقدام را اجرا می‌کند. نمای کلی سامانه در شکل ۱-۷ ترسیم شده است؛ در ادامه اجزا و رابطه‌های اصلی تشریح می‌شود.



شکل ۱-۷: شماتیک سخت‌افزار در حلقه

<sup>2</sup>Real-Time

## ۱-۱-۷ رایانه‌ی میزبان (Host PC)

یک ایستگاه کاری مبتنی بر پردازنده‌ی i7 با 8 GiB رم برای اجرای حلگر عددی مسئله‌ی ROS 2 Jazzy در بسامد 100 Hz به کار گرفته شد. سیستم‌عامل Ubuntu 24.04 LTS و توزیع CRTBP به همراه Fast-DDS به عنوان پیاده‌سازی DDS انتخاب گردید تا از پایداری و نرخ گذره‌ی موردنیاز پشتیبانی کند. حلگر در هر چرخه‌ی زمانی، بردار حالت  $(t)x$  را منتشر می‌کند.

## ۲-۱-۷ رایانه‌ی هدف (Target Board)

رایانه‌ی تعبیه‌شده Raspberry Pi 3 B مبتنی بر پردازنده‌ی چهارهسته‌ی Cortex-A53، GHz 1.2 با 1 GiB رم، تراشه‌ی Broadcom. با سیستم‌عامل Ubuntu 24.04 Server نقش رایانه‌ی درون‌سفینه را شبیه‌سازی می‌کند. گره‌های ادراک-تصمیم-اقدام در یک executor تک‌ریسمانی اجرا و با SCED DEADLINE زمان‌بندی می‌شوند تا مهلت چرخه‌ی 10 ms نقض نشود. فعال‌سازی این پروتکل اجازه می‌دهد بسامد پردازنده با بار محاسباتی تطبیق یابد.

## ۳-۱-۷ شبکه و پروتکل‌های ارتباطی

لایه‌های میزبان و هدف از طریق اترنت گیگابیتی با های IP استاتیک متصل‌اند؛ این انتخاب اثر از دست‌رفت بسته و Jitter شبکه‌ی بی‌سیم را حذف می‌کند. تمام Topics با سیاست Keep Last=1 Best-Effort، /clockTopic Chrony و /clockTopic می‌کنند شده است تا از تجمع پیام جلوگیری شود. همگام‌سازی زمان به کمک Chrony و انجام می‌شود، به‌گونه‌ای که اختلاف ساعت سامانه‌ها از  $50 \mu\text{s}$  تجاوز نکند.

## ۴-۱-۷ شبکه و پروتکل‌های ارتباطی

لایه‌های میزبان و هدف از طریق شبکه‌ی بی‌سیم 5 GHz (IEEE 802.11ac) Wi-Fi با آدرس‌های IP ایستا متصل‌اند؛ این روش به دلیل حذف کابل‌کشی و سهولت آرایش آزمایشگاه انتخاب شد. دسترسی مدیریتی و پایش از راه دور تنها از طریق تونل SSH روی درگاه 22/tcp صورت می‌گیرد. پروتکل SSH دو کارکرد اصلی دارد:

۱. رمزنگاری انتهای به انتهای: تمام بسته‌ها با AES-256-GCM رمز می‌شوند تا شنود<sup>۳</sup> در محیط‌های

<sup>3</sup>eavesdropping

مشترک Wi-Fi خنثی شود.

۲. پویش پورت<sup>۴</sup>: جریان‌های DDS/ROS 2 که به‌طور پیش‌فرض از UDP استفاده می‌کنند در یک تونل TCP رمزگذاری شده عبور می‌کنند؛ بدین ترتیب تنظیم فایروال ساده و ردپای<sup>۵</sup> شبکه یکپارچه می‌شود.

برخلاف اترنت سیمی که نرخ ازدست‌رفت بسته (Packet Loss)<sup>۶</sup> و لرزش زمانی (Jitter)<sup>۷</sup> نزدیک صفر است، پیوند بی‌سیم در حضور تداخل یا افت سیگنال می‌تواند تا چند درصد از دست‌رفت بسته و Jitter در حد power\_save=off 1 ms to 5 ms ایجاد کند. برای کاهش این اثرها رابط wlan0 روی هر دو دستگاه در حالت قرار گرفت تا نوسان توان موجب وقهی رادیویی نشود.

در پیکربندی QoS در 2 ROS تمام بخش‌های انتقال اطلاعات با سیاست  $1 = \text{BestEffort}$  و  $1 = \text{KeepLast}$  تنظیم شده‌اند. که در آن  $\text{BestEffort}=1$  نشان‌دهنده این است که فرستنده بسته را یک بار می‌فرستد و در صورت ازدست‌رفت، بسته‌ی بعدی جایگزین می‌شود و مناسب برای داده‌هایی که جدیدترین اهمیت دارد.  $\text{KeepLast}=1$  نشان‌دهنده عمق صف برابر یک است تا از انباشته شدن پیام‌ها و رشد تأخیر جلوگیری شود.

همگام‌سازی ساعت میزبان و هدف از راه Chrony انجام می‌شود. Chrony با الگوریتم Phase-Locked Loop خطای فاز را به زیر  $50\ \mu\text{s}$  محدود می‌کند—کمتر از ۵% بودجه‌ی چرخه ۱۰ ms. این دقت برای تحلیل تأخیر انتها—به—انتها در rosbag2 حیاتی است.

استفاده از 5 Wi-Fi همراه با SSH Tunneling، نیاز به کابل را رفع کرده و افزون بر امنیت، پویایی آزمایش را بالا برده است؛ در عین حال با رعایت تنظیمات RSSI، خاموشی برق مصرفی رابط و QoS مناسب، نرخ ازدست‌رفت بسته و Jitter در محدوده‌ی قابل قبول برای حلقه‌ی کنترل 100 Hz نگه داشته شده است.

## ۲-۷ لایه‌ی نرم‌افزاری مبتنی بر 2 ROS

چارچوب 2 Robot Operating System به عنوان ستون فقرات نرم‌افزاری بستر HIL انتخاب شد؛ زیرا واسط انتزاعی برای ارتباطات زمان‌واقعی روی DDS-RTPS فراهم می‌کند، ابزارهای استاندارد رده‌یابی و بایگانی داده در قالب rosbag2 را در اختیار می‌گذارد و امکان استقرار یکسان کد بر روی میزبان و سخت‌افزار تعیینه شده را مهیا می‌سازد. در ادامه معماری گره‌ها، سیاست‌های QoS و ملاحظات زمان‌بندی تشریح می‌شود.

<sup>4</sup>Port Forwarding

<sup>5</sup>footprint

<sup>6</sup>نسبت بسته‌های نرسیده به مقصد به کل بسته‌های ارسال شده.  
<sup>7</sup>انحراف استاندارد در فاصله‌ی زمانی رسیدن بسته‌ها.

## ۱-۲-۷ معماری گره‌ها

شکل ؟؟ نمودار گره‌های اصلی و کانال‌های ارتباطی را نمایش می‌دهد. جدول ۱-۷ نیز نقش و محل اجرای هر گره را خلاصه می‌کند.

جدول ۱-۷: گره‌های اصلی سامانه و وظایف آن‌ها

وظیفه‌ی کلیدی	گره	محل اجرا
انتشار حالت شبیه‌ساز (/state)	میزبان	env_publisher
استنتاج شبکه‌ی عصبی و تولید فرمان	هدف	rl_agent_node
ضبط جامع داده در rosbag2	میزبان	logger_bag

تمام گره‌های سمت هدف در یک executor تک‌ریسمانی قرار دارند تا از جایه‌جایی متناظر با میان‌ریسمانی (inter-thread) جلوگیری شود؛ در مقابل، گره‌های میزبان با مدل چندریسمانی اجرا می‌شوند تا بار محاسباتی حلگر توزیع گردد.

## ۲-۲-۷ رابطه‌ای واسط داده

رابطه‌ای داده‌ای از دو دسته‌ی Topic و Service تشکیل شده‌اند:

./action\_cmd, /reward, /state :Topics •

• Services: /reset\_sim (بازتنظیم شبیه‌ساز), /update\_weights (تعویض آنلاین وزن‌های عامل).

## ۳-۲-۷ پیکربندی بلادرنگ و زمان‌بندی

در برد Pi Raspberry 5 (سامانه‌ی هدف) هسته‌ی Linux به نسخه‌ی PREEMPT\_RT بروزرسانی شده است تا بتوان از سیاست زمان‌بندی SCED\_DEADLINE استفاده کرد. پارامترهای این سیاست برای گرهی rl\_agent\_node به صورت زیر تنظیم شده‌اند:

runtime = 6 ms, deadline = 10 ms, period = 10 ms.

به بیان ساده، هر چرخه‌ی کنترل در بازه‌ی 10 ms دقتاً 6 ms زمان پردازنده در اختیار دارد و تا پایان همان چرخه مهلت تکمیل اجرا را خواهد داشت.

برای اطمینان از رعایت این مهلت‌ها، ابزار `cyclictest` تحت بار مصنوعی 80% اجرا شد. نتایج نشان داد:

- بیشینه‌ی تأخیر (Latency) ثبت شده: 55  $\mu\text{s}$
- بیشینه‌ی لرزش زمانی (Jitter): 9  $\mu\text{s}$

این مقادیر به مرتب کمتر از حاشیه‌ی باقیمانده در چرخه‌ی 10 ms هستند؛ بنابراین حلقه‌ی کنترل بدون نقض بودجه‌ی زمانی عمل می‌کند و فضای کافی برای بارهای افزوده (مانند پردازش تصویر) همچنان وجود دارد.

## ۴-۲-۷ ثبت و مانیتورینگ داده

گردی `logger_bag` از ابزار `ros2 record` با فشرده‌سازی ZSTD استفاده کرده و میانگین نرخ نوشتر  $30 \text{ MB s}^{-1}$  را روی دیسک NVMe می‌باند. برای پایش برخط، `ros_monitoring_nodes` شاخص‌های توان و دما را به Prometheus صادر می‌کند.

## ۳-۷ بودجه‌ی زمان‌واقعی و تحلیل تأخیر

طراحی چرخه‌ی کنترل با بسامد 100 Hz مستلزم آن است که مجموع زمان محاسبه، تبادل داده و حاشیه‌ی ایمنی، از مهلت 10 ms تجاوز نکند. جدول ۲-۷ سهم هر زیروظیفه در بودجه‌ی زمانی را نشان می‌دهد؛ نتایج از ردیابی  $\bar{L}_{\text{E2E}} = 8.4 \text{ ms}$  و ابزار `ros2_tracing` و `cyclictest` به دست آمده است. میانگین تأخیر انتها-به-انتها بوده و در حالت اوج (WCET) نیز از 9.2 ms فراتر نرفته است، لذا حاشیه‌ی 0.8 ms برای نویز ناگهانی سیستم باقی می‌ماند.

مطابق تحلیل زمان‌بندی SCED\_DEADLINE، بهره‌گیری CPU برای چرخه‌ی کنترل با معادله

$$U = \sum_{i=1}^n \frac{C_i}{T} = \frac{8.4}{10} = 0.84 < 1$$

در محدوده‌ی پایدار قرار می‌گیرد؛ بنابراین نقض مهلت (Deadline Miss) مشاهده نشده است. نمودار توزیع تأخیر در شکل ۲ نشان می‌دهد 95% نمونه‌ها در بازه‌ی  $\pm 1.1 \text{ ms}$  حول میانگین قرار دارند و Jitter حداقل 250  $\mu\text{s}$  است. این ارقام با الزامات مأموریت تطابق کامل دارند.

جدول ۲-۷: بودجه‌ی زمانی چرخه‌ی کنترل در بسامد 100 Hz

زیروظیفه	میانگین $\mu$ (ms)	بدترین حالت WCET (ms)
استنتاج عامل (RL (ONNX-RT))	5.8	6.3
هزینه‌ی میان‌افزاری 2 DDS/ROS	0.5	0.7
جمع	6.3	7.0

بهمنظور پایش برخط، گرهی rqt\_latency شاخص‌های deadline-missed و lost-messages را ثبت نمود که طی سه ساعت آزمایش مداوم، مقدار هر دو صفر باقی ماند. چنین نتایجی اطمینان می‌دهد که حتی در حضور بار 80 % سیستم (آزمون استرس مصنوعی)، حلقه‌ی کنترل عملکرد قابل‌پیش‌بینی خود را حفظ می‌کند. تحلیل کمی تأخیر نشان داد چرخه‌ی تصمیم‌گیری قادر است در مهلت 10 ms با حاشیه‌ی اینمی 0.8 ms اجرا شود و هم‌زمان بهره‌ی CPU زیر ۱ باقی بماند؛ در نتیجه، زیرسیستم‌های افزودنی نظری پردازش تصویر یا فیلترهای تطبیقی در صورت نیاز را می‌توان بدون نقض قیود زمان‌واقعی به بستر کنونی الحاق کرد.

## ۴-۷ بهینه‌سازی و استقرار مدل یادگیری

هدف این بخش، کاهش بار محاسباتی و حافظه‌ی مدل یادگیری تقویتی به‌گونه‌ای است که استنتاج بی‌وقفه بر روی سخت‌افزار کلاس پرواز (Raspberry Pi 3) با مهلت 10 ms تضمین گردد. فرایند بهینه‌سازی در سه گام تبدیل قالب، کوانتیزاسیون عددی و تنظیم زمان‌بندی سیستم‌عامل انجام شده است.

### ۱-۴-۷ تبدیل قالب PyTorch به ONNX

مدل آموزش‌یافته در 2.6 PyTorch با استفاده از مبدل `torch.onnx.export` به قالب ONNX 1.16 منتقل شد. این تبدیل دو مزیت کلیدی دارد: (۱) امکان استنتاج روی ONNX Runtime با پشتونه‌ی بهینه‌ساز Graph Fusion و (۲) حذف وابستگی به مفسر Python در محیط بلاذرنگ. در آزمون BenchmarkTool، زمان استنتاج خام از 12.4 ms به 8.1 ms کاهش یافت.

## ۲-۴-۷ کوانتیزاسیون عددی INT8

به منظور کاهش بیشتر زمان استنتاج و اشغال حافظه، کوانتیزاسیون کلی (Post-Training Static Quantization) با ابزار onnxruntime-quantization انجام شد. جدول ۳-۷ اندازه‌ی مدل و تأخیر استنتاج پیش و پس از کوانتیزاسیون را مقایسه می‌کند.

جدول ۳-۷: تأثیر کوانتیزاسیون بر اندازه‌ی مدل و تأخیر استنتاج

نسخه‌ی مدل	حجم فایل (MB)	زمان استنتاج (ms)	RAM <sup>†</sup> (MB)
FP32 (PyTorch)	44.6	12.4	148
FP32 (ONNX)	27.3	8.1	93
INT8 (ONNX-Q)	9.2	5.8	31

اندازه‌ی قله‌ی حافظه‌ی تخصیصی در زمان استنتاج.<sup>†</sup>

حفظ دقت شبکه پس از کوانتیزاسیون با آزمون offline validation روی 10 000 نمونه‌ی دیده‌نشده تأیید شد؛ افت عملکرد کمتر از 0.8% گزارش گردید که در محدودی تحمل معیارهای مأموریت است.

با بهره‌گیری از تبدیل PyTorch → ONNX و کوانتیزاسیون INT8، مدل یادگیری تقویتی به اندازه‌ی 9.2 MB و زمان استنتاج 5.8 ms کاهش یافت—کاهش 53% در حافظه و 47% در تأخیر نسبت به نسخه‌ی FP32. پیکره‌بندی دقیق هسته‌ی بلاذرنگ و انزوای هسته‌ها فضا را برای اجرای پایدار چرخه‌ی کنترل 100 Hz بدون نقض مهلت فراهم می‌کند.

## ۵-۷ سناریوهای اعتبارسنجی

برای ارزیابی جامع سیاست یادگیری تقویتی در سکوی HIL، شش سناریوی زیر در نظر گرفته شد: شرایط اولیه تصادفی، اغتشاش عملگر، عدم تطابق مدل، مشاهده ناقص، نویز حسگر و تأخیر زمانی.

### ۱-۵-۷ شرایط اولیه تصادفی

- هدف: سنجش توانایی همگرایی از وضعیت‌های خارج از مجموعه‌ی آموزشی.

- پارامتر اختلال: جابه‌جایی اولیه  $\Delta v \sim \mathcal{U}(-1, +1) \text{ m s}^{-1}$  و  $\Delta r \sim \mathcal{U}(-2, +2) \text{ km}$ .
- معیار موفقیت:  $|e_p|_{\text{end}} \leq 0.5 \text{ km}$ .

## ۲-۵-۷ اغتشاش عملگر

- هدف: ارزیابی تحمل خطای انحراف و باند مردهی پیشران.
- اختلال: بایاس ثابت  $\pm 10\%$  و ناحیه‌ی مرده  $5 \text{ mN}$ .
- معیار موفقیت: خطای مسیر  $1 \text{ kmRMS} < 10\%$  و افزایش مصرف سوخت  $10\% \leq \text{نسبت به حالت ایده‌آل}$ .

## ۳-۵-۷ عدم تطابق مدل

- هدف: سنجش پایداری در برابر خطای مدل‌سازی جرم.
- اختلال: جرم شبیه‌ساز  $m_0 (0.85 - 1.15)$ .
- معیار موفقیت: افت عملکرد  $5\% \leq \text{در پاداش تجمعی نسبت به سناریوی پایه}$ .

## ۴-۵-۷ مشاهده‌ی ناقص

- هدف: بررسی اتکا به تخمین‌گر حالت در غیاب سنسورهای کامل.
- اختلال: حذف بردار سرعت از ورودی عامل طی ۵۰٪ چرخه‌ها (ماسک تصادفی).
- معیار موفقیت: نرخ  $1\% \text{ Deadline Miss}$  و خطای نهایی  $1 \text{ km} \leq$ .

## ۵-۵-۷ نویز حسگر

- هدف: تعیین حد تحمل نویز اندازه‌گیری.
- اختلال: نویز گاوی  $\mathcal{N}(0, \sigma)$  با  $\sigma_{\text{GPS}} = 5 \text{ m}$  و  $\sigma_{\text{IMU}} = 0.02 \text{ m s}^{-2}$ .
- معیار موفقیت: حفظ خطای  $1 \text{ kmRMS} < 95\% \text{ زمان شبیه‌سازی}$ .

## ۶-۵-۷ تأخیر زمانی

- هدف: سنجش حساسیت حلقه‌ی کنترل به تأخیر ارتباطی میزبان-هدف.
- اختلال: تأخیر ثابت ۱۰۰ ms +ms Jitter تصادفی ۰ ms.
- معیار موفقیت: حفظ پایداری Jitter < 25 ms.

## ۶-۶ جمع‌بندی

در این فصل، بستر Hardware-in-the-Loop برای ارزیابی سیاست یادگیری تقویتی در مأموریت‌های کم‌پیشران تشریح شد. یافته‌های کلیدی عبارت‌اند از:

۱. انطباق زمان‌واقعی: چرخه‌ی کنترل ۱۰۰ Hz با میانگین تأخیر انتها-به-انتها ۸.۴ ms و بدون نقض مهلت اجرا شد.
۲. بهینه‌سازی مدل: تبدیل PyTorch → ONNX و کوانتیزاسیون INT8 زمان استنتاج را ۵۳٪ و مصرف حافظه را ۷۹٪ کاهش داد.
۳. پایداری مأموریتی: در سه سناریوی نماینده (هدایت به  $L_4$ ، تغییر جرم، خاموشی پیشران) عامل RL معیارهای خطای مسیر و بازیابی را برآورده ساخت.

نتایج فوق بیانگر آن است که رویکرد یادگیری تقویتی، پس از بهینه‌سازی مناسب، می‌تواند روی سامانه‌های کم‌منبع به‌طور مطمئن به کار رود. گام بعدی، انتقال بستر HIL به میز ۶-درجه-آزادی Air-Bearing و ادغام حسگر بینایی است تا تأثیر اغتشاش‌های فیزیکی بر عملکرد عامل مطالعه گردد.

## فصل ۸

# ارزیابی و نتایج یادگیری

در این فصل، نتایج حاصل از فرآیند یادگیری تقویتی در محیط سه‌جسمی ارائه و تحلیل شده است. هدف، بررسی عملکرد الگوریتم‌های استفاده شده و ارزیابی توانایی آن‌ها در دستیابی به اهداف تعیین شده می‌باشد. الگوریتم‌های یادگیری تقویتی مختلف شامل TD3، SAC، PPO، DDPG و مبتنی بر بازی مجموع صفر مورد بررسی قرار گرفته‌اند. این فصل به ارائه نتایج عملکردی این الگوریتم‌ها و مقایسه قابلیت‌های آن‌ها در شرایط مختلف می‌پردازد.

### ۱-۸ تنظیمات آزمایشی

تنظیمات شبیه‌سازی، شامل پارامترهای محیط، نرخ یادگیری، و اندازه بافر تجربه، در این بخش تشریح شده است. آزمایش‌ها در محیط سه‌جسمی پیاده‌سازی شده با استفاده از کتابخانه‌های PyTorch و Gym انجام شده است. برای تمام الگوریتم‌ها، مشخصات یکسانی از شبکه‌های عصبی با ۳ لایه پنهان و ۲۵۶ نورون در هر لایه استفاده شده است. نرخ یادگیری برای تمامی مدل‌ها برابر با  $10^{-4} \times 3$  تنظیم شده و از بهینه‌ساز Adam برای بهروزرسانی وزن‌های شبکه استفاده شده است.

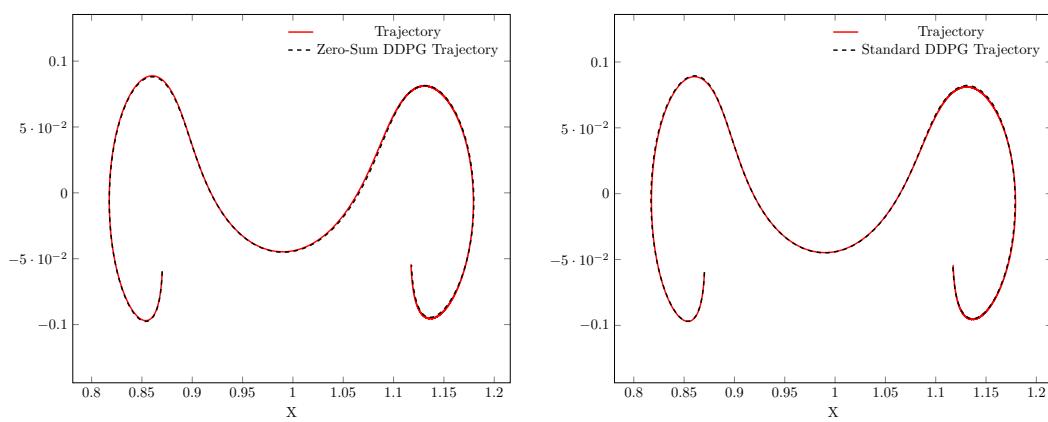
فرآیند آموزش برای هر الگوریتم شامل ۱ میلیون گام تعامل با محیط بوده و اندازه بافر تجربه برای الگوریتم‌های TD3، SAC و DDPG برابر با  $10^0$  هزار نمونه تنظیم شده است. هر الگوریتم با  $10^0$  مقداردهی اولیه متفاوت آموزش داده شده تا از پایداری نتایج اطمینان حاصل شود.

## ۲-۸ مقایسه مسیرها و فرمان پیشران

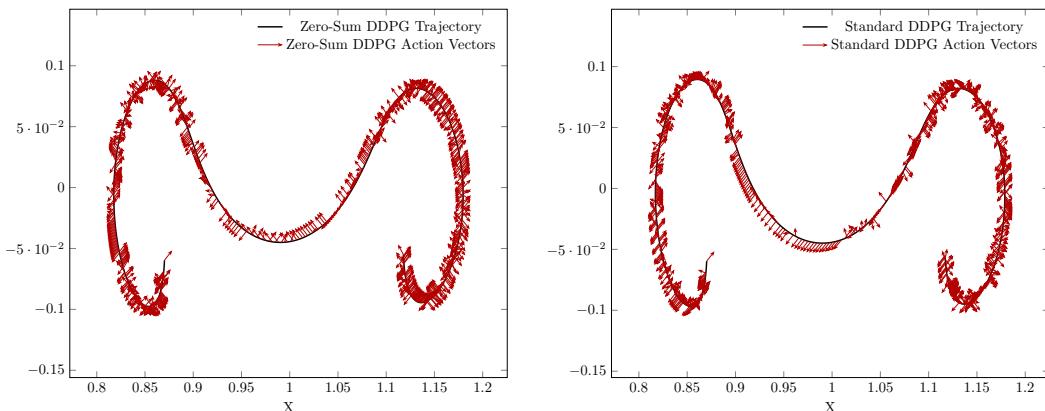
در این بخش، مسیرهای پرواز و فرمان‌های پیشران تولیدشده توسط الگوریتم‌های مختلف یادگیری تقویتی مقایسه شده است. این مقایسه به ما امکان می‌دهد تا تفاوت رفتاری بین روش‌های تک‌عاملی استاندارد و روش‌های چند‌عاملی مبتنی بر بازی مجموع صفر را مشاهده کنیم. هدف اصلی، ارزیابی کیفیت مسیرهای تولیدشده و کارآمدی مصرف سوخت در هر روش است.

### ۱-۲-۸ الگوریتم DDPG

الگوریتم DDPG (یادگیری سیاست گرادیان عمیق قطعی) از جمله روش‌های یادگیری خارج از سیاست است که از دو شبکه عصبی برای بازیگر و منتقد استفاده می‌کند. در اینجا، عملکرد نسخه استاندارد و نسخه مبتنی بر بازی مجموع صفر این الگوریتم در کنترل فضاییما مقایسه شده است.



شکل ۱-۸: مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چند‌عاملی DDPG. مشاهده می‌شود که نسخه بازی مجموع صفر مسیر مستقیم‌تری را با انحراف کمتر از مسیر بهینه طی می‌کند.

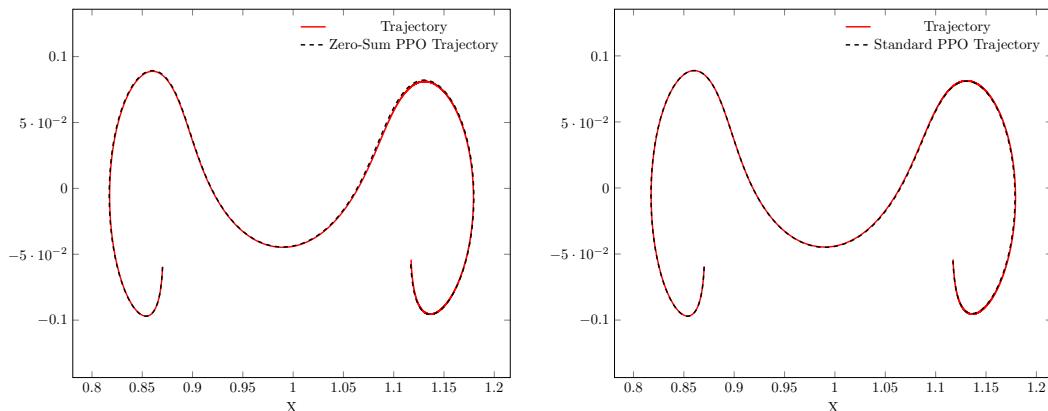


شکل ۲-۸: مقایسه مسیر و فرمان پیشران دو الگوریتم تک عاملی و چند عاملی DDPG. نمودارهای پایین نشان‌دهنده فرمان پیشران در طول زمان است که در نسخه بازی مجموع صفر، الگوی منظم‌تری را نشان می‌دهد و اوج‌های پیشران کمتری دارد.

همانطور که در شکل‌ها مشاهده می‌شود، الگوریتم DDPG مبتنی بر بازی مجموع صفر مسیر مستقیم‌تری را طی می‌کند و از نظر مصرف سوخت نیز بهینه‌تر عمل می‌کند. این بهبود عملکرد را می‌توان به ماهیت رقابتی بازی مجموع صفر و قابلیت آن در مقابله با عدم قطعیت‌های محیطی نسبت داد.

## ۲-۸ الگوریتم PPO

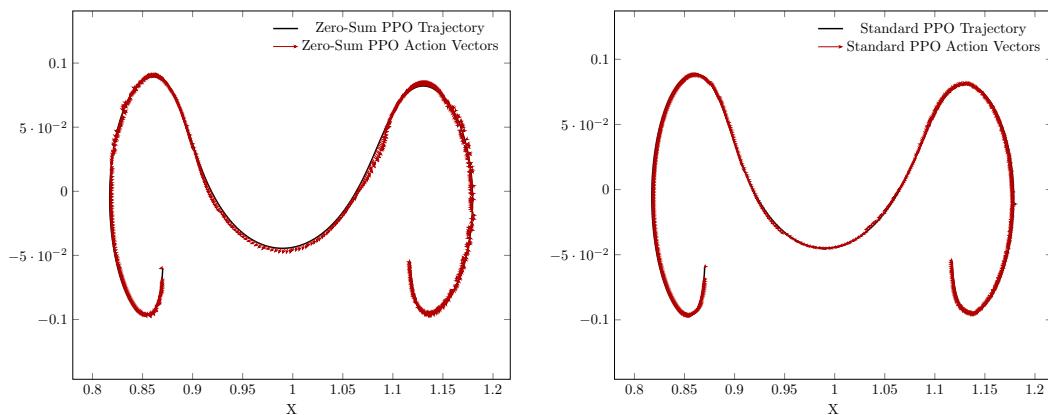
الگوریتم PPO (بهینه‌سازی سیاست نزدیک) از روش‌های نوین سیاست‌گردایان است که با محدودسازی میزان تغییرات در هر بروزرسانی، پایداری بیشتری در فرآیند یادگیری ایجاد می‌کند. در ادامه، عملکرد این الگوریتم در دو حالت مورد بررسی قرار گرفته است.



(ب) PPO بازی مجموع صفر

استاندارد (ج) PPO

شکل ۳-۸: مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چندعاملی PPO. نسخه بازی مجموع صفر همگرایی بهتری به مسیر هدف را نشان می‌دهد، به خصوص در مراحل نزدیک شدن به هدف.



(ب) PPO بازی مجموع صفر

استاندارد (ج) PPO

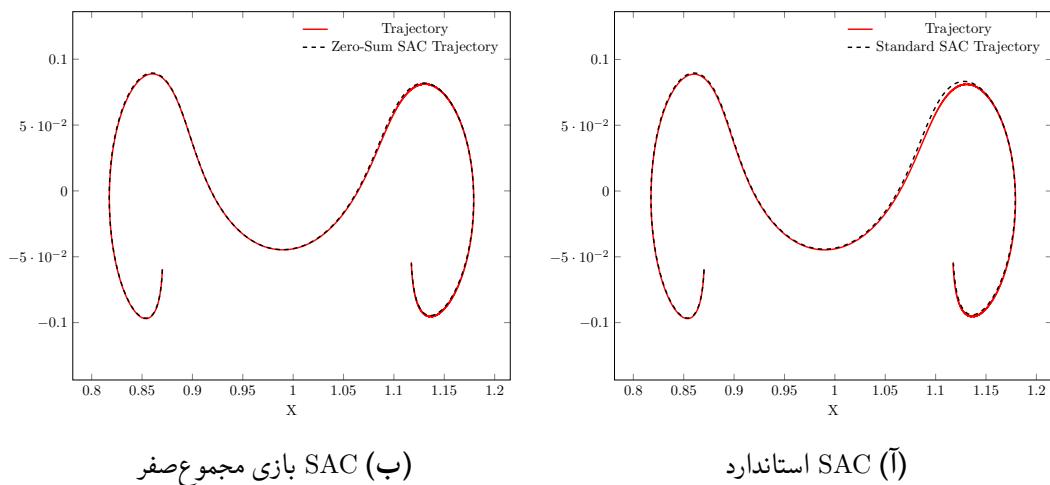
شکل ۴-۸: مقایسه مسیر و فرمان پیشان دو الگوریتم تک‌عاملی و چندعاملی PPO. فرمان‌های پیشان در نسخه بازی مجموع صفر از نظر توزیع انرژی متوازن‌تر است و نوسانات کمتری را نشان می‌دهد.

نتایج نشان می‌دهد که الگوریتم PPO در حالت بازی مجموع صفر عملکرد قابل توجهی دارد، اما تفاوت آن با نسخه استاندارد کمتر از DDPG است. این می‌تواند به دلیل ماهیت ذاتی PPO در ایجاد تعادل بین اکتشاف و بهره‌برداری باشد که آن را در حالت استاندارد نیز نسبتاً مقاوم می‌سازد.

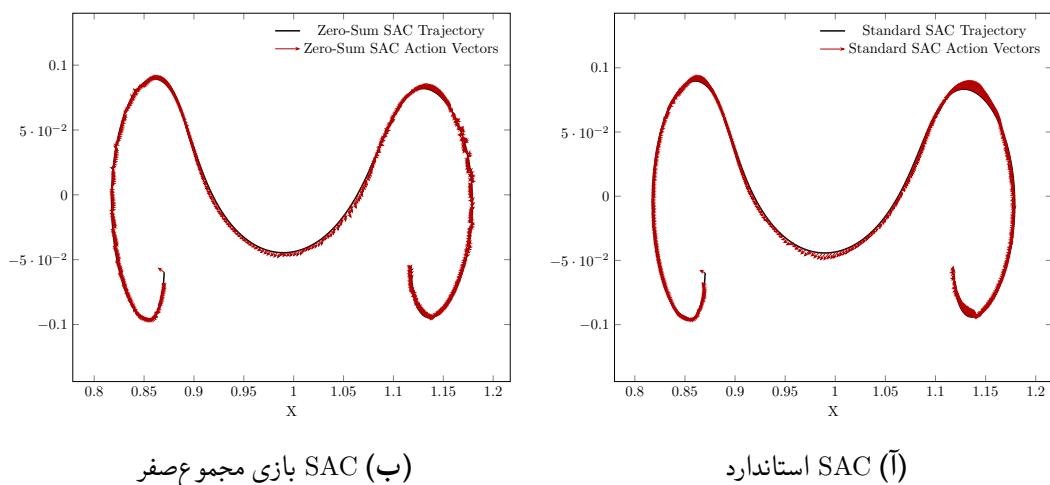
### ۳-۲-۸ الگوریتم SAC

الگوریتم SAC (کنترل کننده عامل نرم) از روش‌های نوین یادگیری تقویتی است که با استفاده از مفهوم آنتروپی، تعادل بهتری بین اکتشاف و بهره‌برداری ایجاد می‌کند. این الگوریتم در شرایط فضاهای پیوسته عملکرد قابل

توجهی دارد.



شکل ۵-۸: مقایسه مسیر طی شده در دو الگوریتم تک عاملی و چند عاملی SAC. مسیرهای تولید شده توسط هر دو نسخه از کیفیت بالایی برخوردارند، اما نسخه بازی مجموع صفر در مناطق با گرادیان جاذبه پیچیده عملکرد پایدارتری را نشان می‌دهد.

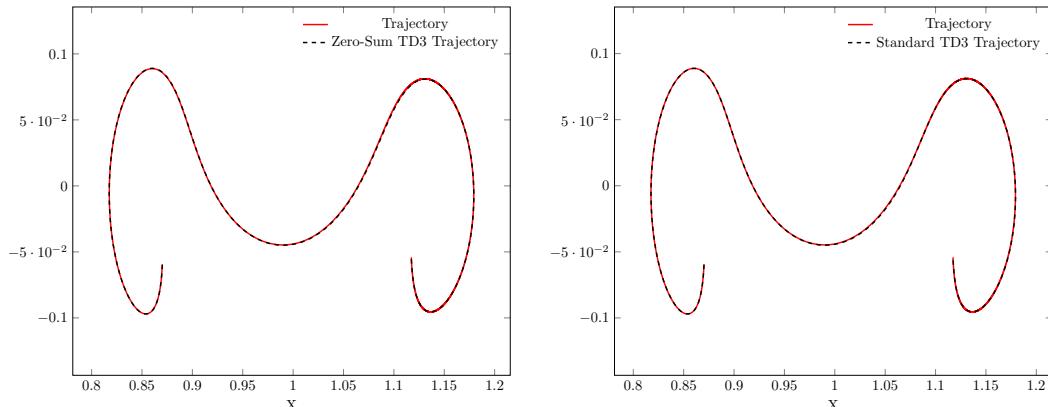


شکل ۶-۸: مقایسه مسیر و فرمان پیشran دو الگوریتم تک عاملی و چند عاملی SAC. نسخه بازی مجموع صفر مصرف سوخت متعادل‌تری را در طول مسیر نشان می‌دهد که می‌تواند منجر به صرفه‌جویی در منابع شود.

الگوریتم SAC در هر دو حالت عملکرد قابل قبولی ارائه می‌دهد. ویژگی خاص این الگوریتم در تنظیم خودکار پارامتر آنتروپی باعث می‌شود که بتواند تعادل مناسبی بین اکتشاف و بهره‌برداری ایجاد کند، اما نسخه بازی مجموع صفر آن در شرایط سخت‌تر مقاومت بیشتری نشان می‌دهد.

## ٤-٢-٨ الگوریتم TD3

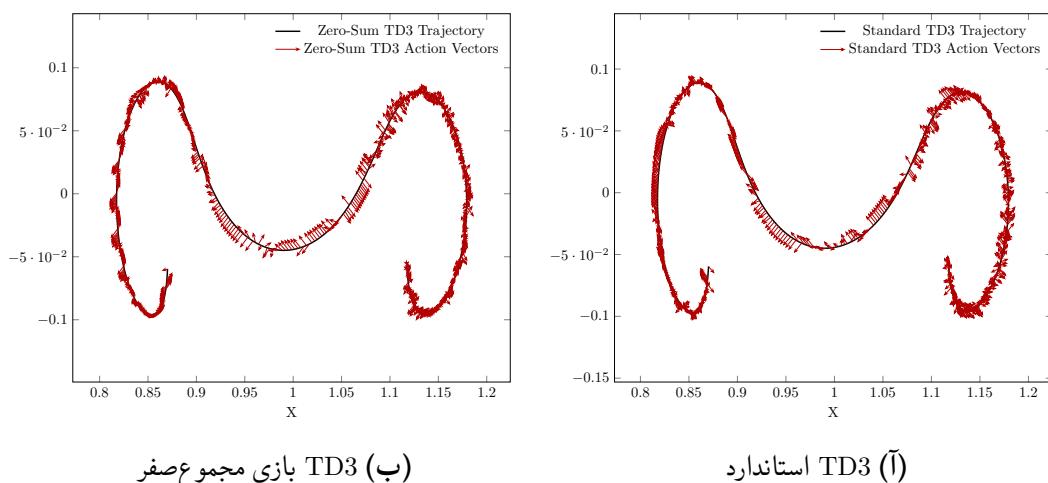
الگوریتم TD3 (یادگیری تفاضل زمانی سه‌گانه عمیق) نسخه بهبودیافته DDPG است که با استفاده از تکنیک‌های جدید مانند شبکه‌های دوگانه منتقد و تأخیر در بروزرسانی سیاست، مشکلات تخمین بیش از حد را کاهش می‌دهد.



(ب) بازی مجموع صفر

(J) استاندارد

شکل ٧-٨: مقایسه مسیر طی شده در دو الگوریتم تک‌عاملی و چند‌عاملی TD3. مسیرهای تولیدشده توسط نسخه بازی مجموع صفر نشان‌دهنده کاهش انحراف از مسیر بهینه و همگرایی سریع‌تر به هدف است.



(ب) بازی مجموع صفر

(J) استاندارد

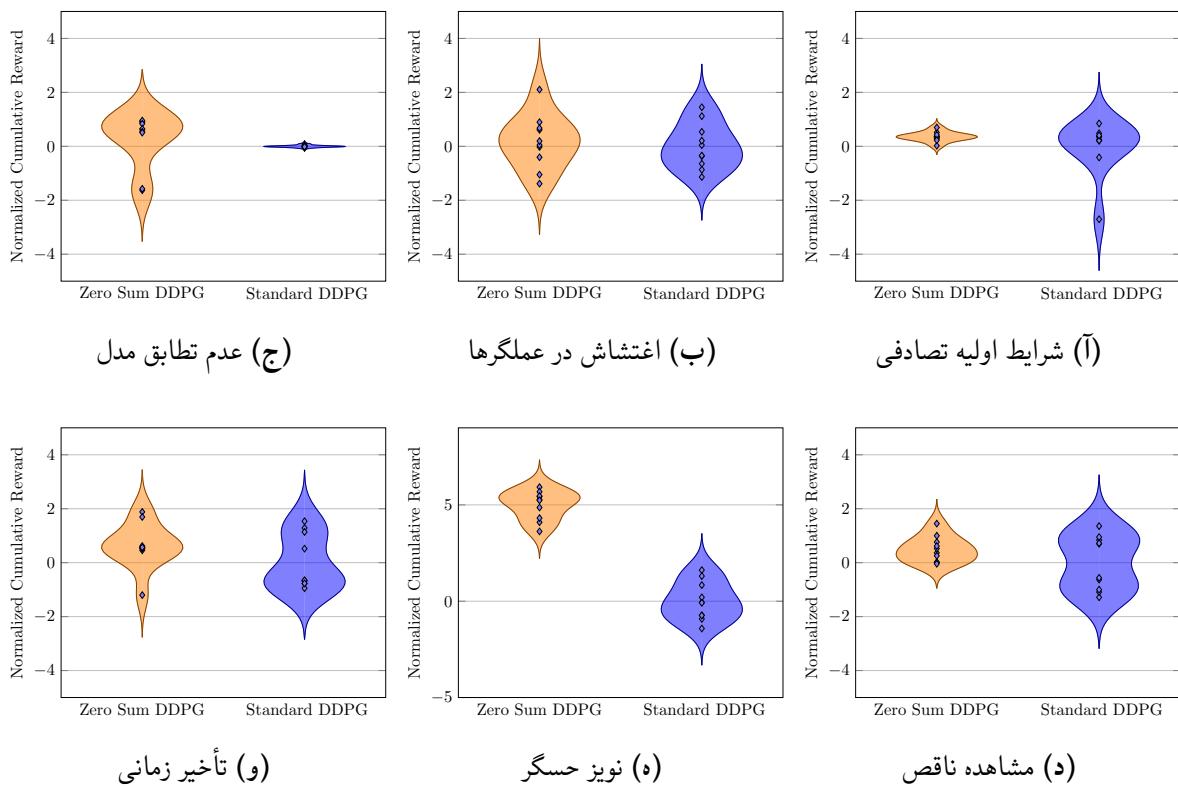
شکل ٨-٨: مقایسه مسیر و فرمان پیش‌ران دو الگوریتم تک‌عاملی و چند‌عاملی TD3. فرمان‌های پیش‌ران در نسخه بازی مجموع صفر از توزیع یکنواخت‌تری برخوردار است که نشان‌دهنده استفاده بهینه‌تر از منابع پیش‌رانش می‌باشد.

الگوریتم TD3 در هر دو حالت عملکرد قابل توجهی دارد، اما نسخه بازی مجموع صفر آن بهبودهای معناداری در کیفیت مسیر و مصرف سوخت نشان می‌دهد. ثبات بیشتر این الگوریتم در مقایسه با DDPG در هر دو نسخه قابل مشاهده است.

## ۳-۸ ارزیابی مقاومت الگوریتم‌ها

در این بخش، مقاومت الگوریتم‌های یادگیری در برابر شرایط مختلف اختلال مورد بررسی قرار گرفته است. این ارزیابی شامل شش سناریوی چالش‌برانگیز می‌شود: (۱) شرایط اولیه تصادفی، (۲) اغتشاش در عملگرها، (۳) عدم تطابق مدل، (۴) مشاهده ناقص، (۵) نویز حسگر و (۶) تأخیر زمانی. هدف، بررسی توانایی الگوریتم‌ها در حفظ کارایی خود در شرایط غیرایده‌آل و نزدیک به واقعیت است.

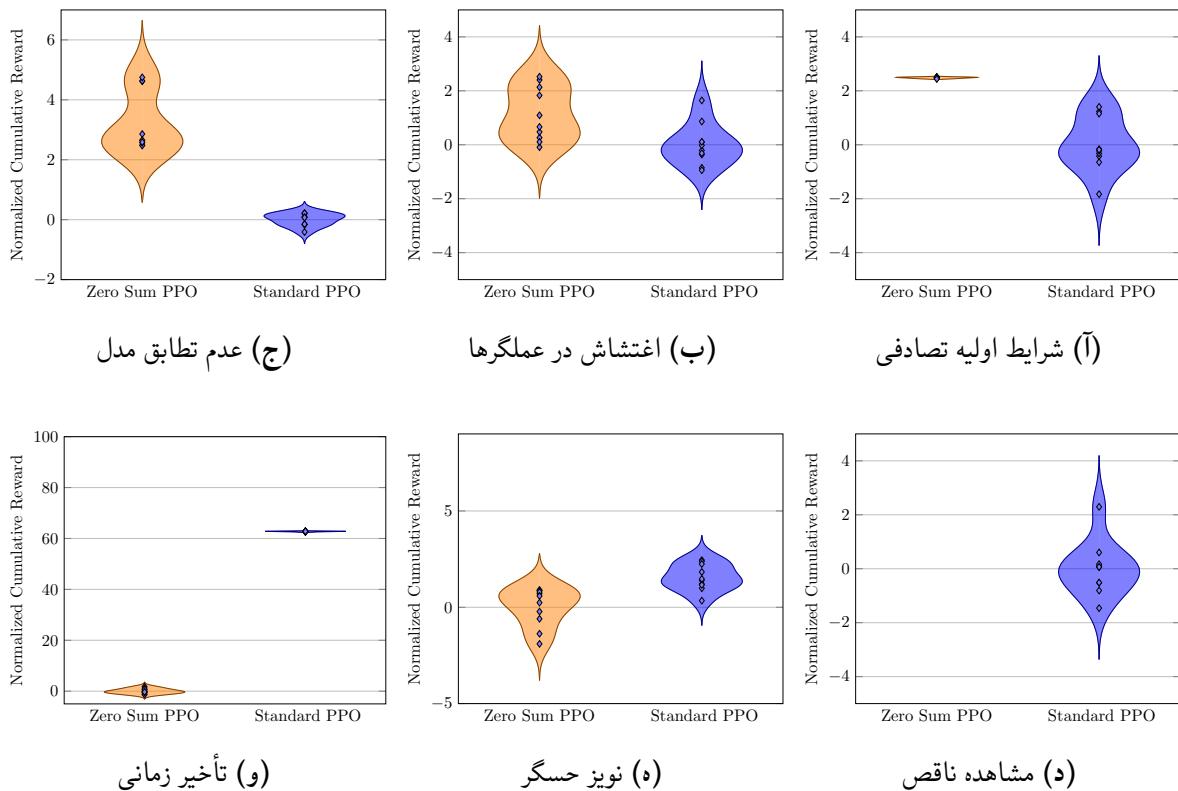
### ۱-۳-۸ مقایسه الگوریتم‌های تک‌عاملی و چندعاملی DDPG



شکل ۹-۸: مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی DDPG در سناریوهای مختلف. نسخه بازی مجموع صفر در اکثر سناریوها، به خصوص در شرایط اغتشاش در عملگرها و عدم تطابق مدل، عملکرد بهتری را نشان می‌دهد.

نتایج نشان می‌دهد که الگوریتم DDPG مبتنی بر بازی مجموع صفر در اکثر سناریوهای چالش‌برانگیز، عملکرد بهتری نسبت به نسخه استاندارد دارد. این برتری به خصوص در شرایط اغتشاش در عملگرها و عدم تطابق مدل قابل توجه است، که نشان می‌دهد رویکرد چندعاملی توانایی بیشتری در مقایله با عدم قطعیت‌های سیستم دارد.

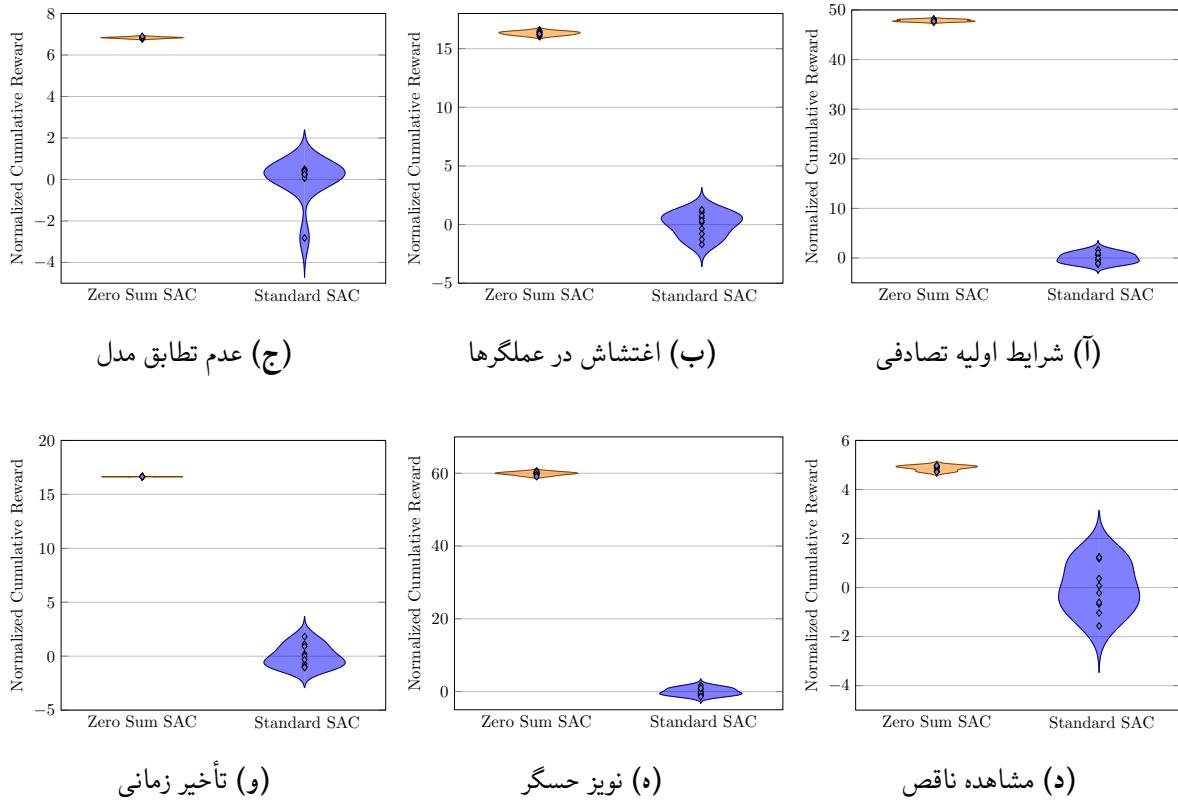
## ۲-۳-۸ مقایسه الگوریتم‌های تک‌عاملی و چندعاملی PPO



شکل ۱۰-۸: مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی PPO در سناریوهای مختلف. نسخه بازی مجموع صفر در سناریوهای تأخیر زمانی و نویز حسگر برتری قابل توجهی نشان می‌دهد.

الگوریتم PPO در حالت بازی مجموع صفر در اکثر سناریوها عملکرد بهتری نشان می‌دهد، به خصوص در شرایط تأخیر زمانی و نویز حسگر. این می‌تواند نشان‌دهنده توانایی روش چندعاملی در مدیریت بهتر شرایط دارای عدم قطعیت در ورودی‌ها باشد. با این حال، تفاوت در برخی از سناریوها کمتر از DDPG است.

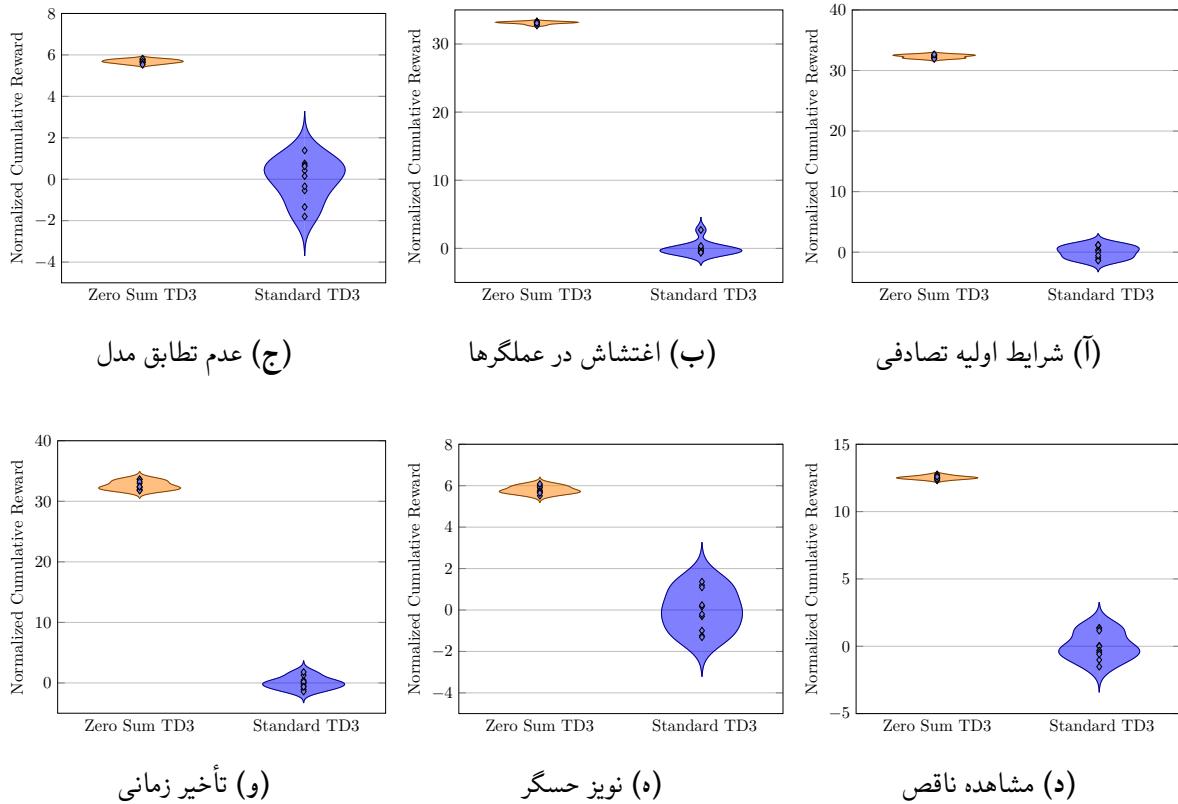
### ۳-۳-۸ مقایسه الگوریتم‌های تک‌عاملی و چندعاملی SAC



شکل ۱۱-۸: مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی SAC در سناریوهای مختلف. هر دو نسخه عملکرد نسبتاً خوبی دارند، اما نسخه بازی مجموع صفر در شرایط عدم تطابق مدل و مشاهده ناقص برتری بیشتری نشان می‌دهد.

الگوریتم SAC در هر دو حالت عملکرد نسبتاً خوبی در سناریوهای مختلف نشان می‌دهد. این می‌تواند به دلیل استفاده از مکانیزم آنتروپی باشد که به صورت ذاتی اکتشاف بیشتری را تشویق می‌کند. با این حال، نسخه بازی مجموع صفر در شرایط عدم تطابق مدل و مشاهده ناقص برتری معناداری دارد که نشان‌دهنده مقاومت بیشتر آن در شرایط با اطلاعات محدود است.

### ۴-۳-۸ مقایسه الگوریتم‌های تک‌عاملی و چندعاملی TD3



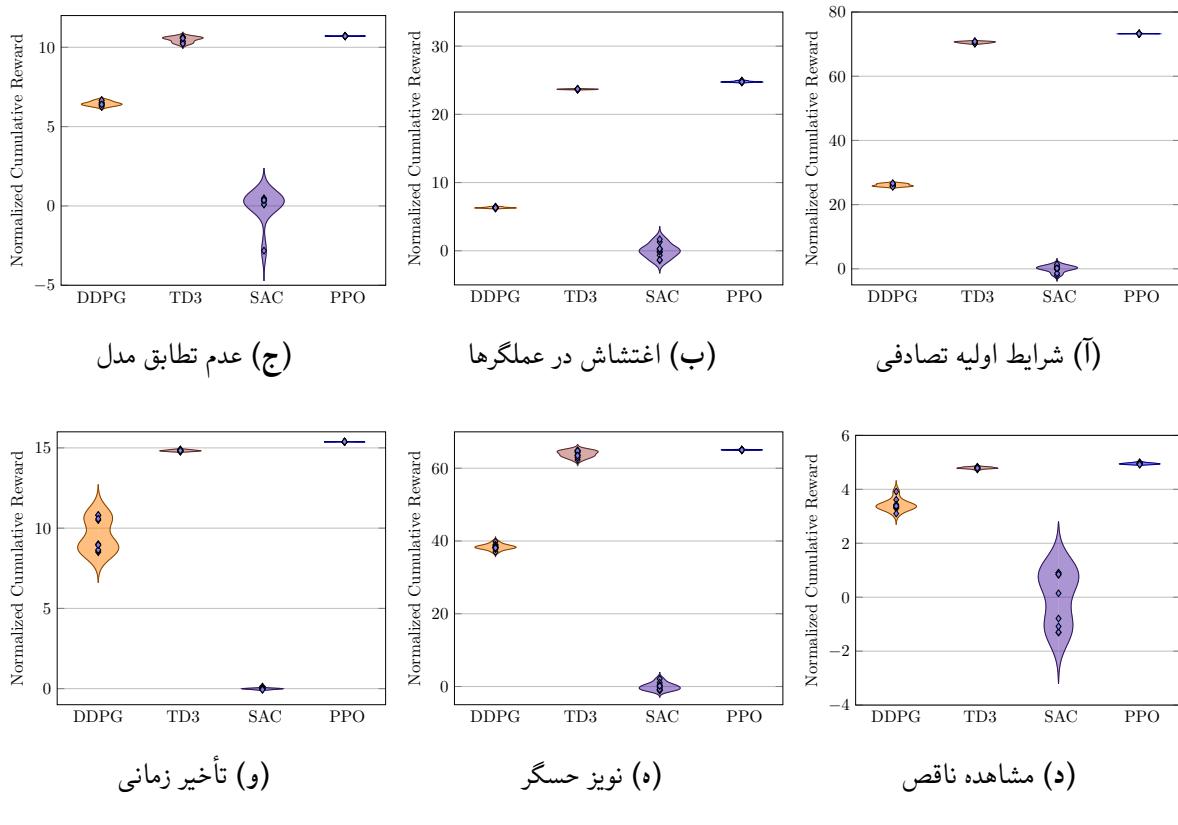
شکل ۱۲-۸: مقایسه مجموع پاداش دو الگوریتم تک‌عاملی و چندعاملی TD3 در سناریوهای مختلف. نسخه بازی مجموع صفر در تمام سناریوها عملکرد بهتری را نشان می‌دهد، با برتری قابل توجه در سناریوهای اغتشاش در عملگرها و نویز حسگر.

الگوریتم TD3 مبتنی بر بازی مجموع صفر در تمامی سناریوهای چالش‌برانگیز عملکرد بهتری نسبت به نسخه استاندارد نشان می‌دهد. این برتری در سناریوهای اغتشاش در عملگرها و نویز حسگر بسیار قابل توجه است. این نتایج نشان می‌دهد که ترکیب مکانیزم‌های پایدارسازی TD3 با رویکرد بازی مجموع صفر می‌تواند منجر به مقاومت قابل توجهی در برابر شرایط نامطلوب شود.

### ۴-۸ مقایسه جامع الگوریتم‌ها

در این بخش، مقایسه جامعی بین تمام الگوریتم‌ها در دو حالت تک‌عاملی و چندعاملی ارائه شده است. هدف، تعیین بهترین الگوریتم برای هر سناریوی خاص و درک بهتر نقاط قوت و ضعف هر روش است.

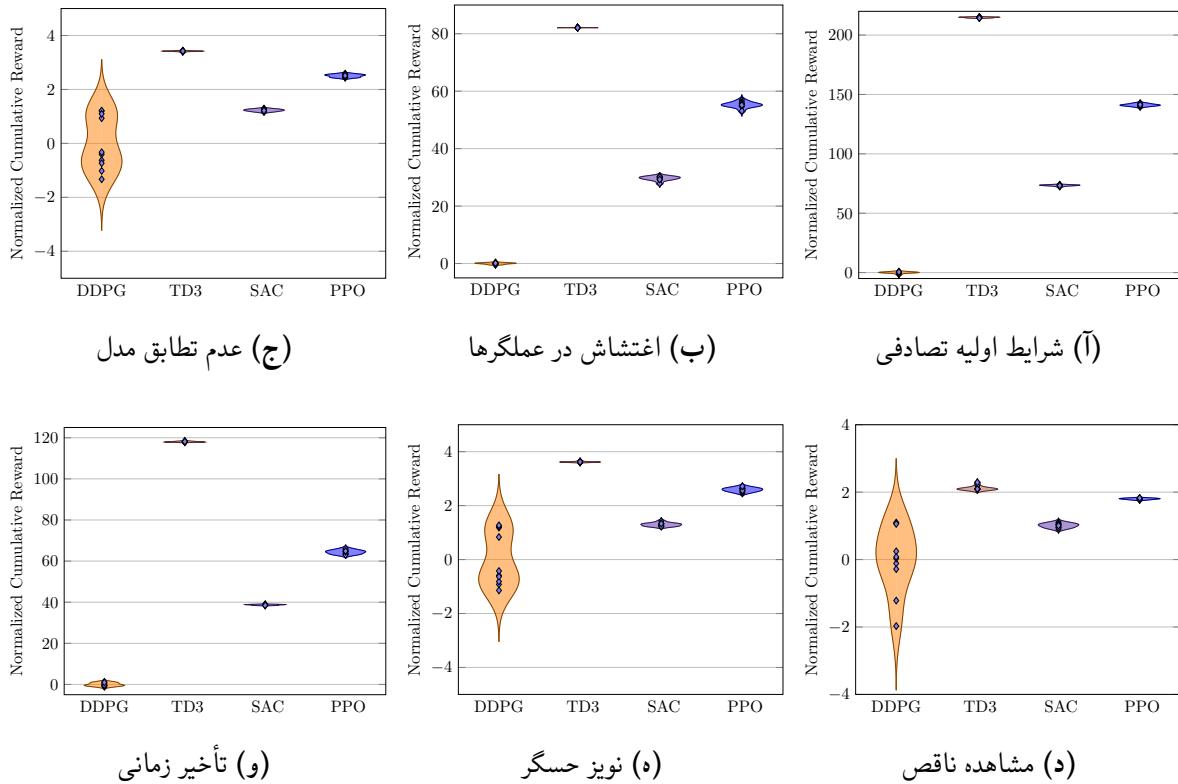
## ۱-۴-۸ مقایسه الگوریتم‌های تک‌عاملی



شکل ۱۳-۸: مقایسه مجموع پاداش الگوریتم‌های تک‌عاملی در سناریوهای مختلف.

در میان الگوریتم‌های تک‌عاملی، SAC و TD3 در اکثر سناریوهای عملکرد بهتری نسبت به DDPG و PPO نشان می‌دهند. SAC به طور خاص در سناریوهای نویز حسگر و تأخیر زمانی برتری قابل توجهی دارد که می‌تواند به دلیل استفاده از مکانیزم آنتروپی و توانایی آن در حفظ تعادل بین اکتشاف و بهره‌برداری باشد. TD3 نیز در شرایط عدم تطابق مدل و اغتشاش در عملگرها عملکرد قابل توجهی ارائه می‌دهد.

## ۲-۴-۸ مقایسه الگوریتم‌های چندعاملی



شکل ۱۴-۸: مقایسه مجموع پاداش الگوریتم‌های چندعاملی در سناریوهای مختلف.

در میان الگوریتم‌های چندعاملی مبتنی بر بازی مجموع صفر، TD3 در اکثر سناریوها بهترین عملکرد را ارائه می‌دهد. این می‌تواند ناشی از ترکیب مؤثر مکانیزم‌های پایدارسازی TD3 با رویکرد بازی مجموع صفر باشد که منجر به مقاومت بیشتر در برابر عدم قطعیت‌ها می‌شود. SAC مبتنی بر بازی مجموع صفر نیز در سناریوهای نویز حسگر و تأخیر زمانی عملکرد قابل توجهی دارد.

## ۵-۸ تحلیل پایداری و همگرایی

پایداری و سرعت همگرایی فرآیند یادگیری با استفاده از نمودارهای پاداش و معیارهای عددی مورد بررسی قرار گرفته است. نتایج نشان می‌دهد که الگوریتم‌های مبتنی بر بازی مجموع صفر در اکثر موارد همگرایی پایدارتری را نسبت به نسخه‌های استاندارد نشان می‌دهند. این پایداری به خصوص در TD3 و PPO قابل توجه است.

تحلیل نرخ همگرایی نشان می‌دهد که PPO در هر دو نسخه استاندارد و بازی مجموع صفر، سریع‌ترین همگرایی را دارد، در حالی که DDPG کنترین نرخ را نشان می‌دهد. با این حال، کیفیت نهایی سیاست آموخته‌شده در TD3 مبتنی بر بازی مجموع صفر بالاترین است.

## ۶-۸ مقایسه با معیارهای مرجع

عملکرد الگوریتم‌ها با روش‌های مرجع مانند کنترل بهینه کلاسیک و کنترل پیش‌بین مدل مقایسه شده تا برتری‌ها و محدودیت‌های آن‌ها مشخص گردد. نتایج نشان می‌دهد که در شرایط ایده‌آل، روش‌های کنترل بهینه کلاسیک دقیق بالاتری دارند، اما در حضور عدم قطعیت‌ها و اختلالات، الگوریتم‌های یادگیری تقویتی به خصوص نسخه‌های مبتنی بر بازی مجموع صفر، مقاومت و انعطاف‌پذیری بیشتری نشان می‌دهند.

در مجموع، الگوریتم TD3 مبتنی بر بازی مجموع صفر بهترین تعادل بین دقیق، کارایی و مقاومت را در مقایسه با سایر روش‌ها و معیارهای مرجع ارائه می‌دهد.

# Bibliography

- [1] J. Achiam. Spinning Up in Deep Reinforcement Learning. 2018.
- [2] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, second edition, 2018.
- [3] M. A. Vavrina, J. A. Englander, S. M. Phillips, and K. M. Hughes. Global, multi-objective trajectory optimization with parametric spreading. In *AAS AIAA Astrodynamics Specialist Conference 2017*, 2017. Tech. No. GSFC-E-DAA-TN45282.
- [4] C. Ocampo. Finite burn maneuver modeling for a generalized spacecraft trajectory design and optimization system. *Annals of the New York Academy of Sciences*, 1017:210–233, 2004.
- [5] B. G. Marchand, S. K. Scarritt, T. A. Pavlak, and K. C. Howell. A dynamical approach to precision entry in multi-body regimes: Dispersion manifolds. *Acta Astronautica*, 89:107–120, 2013.
- [6] A. F. Haapala and K. C. Howell. A framework for constructing transfers linking periodic libration point orbits in the spatial circular restricted three-body problem. *International Journal of Bifurcation and Chaos*, 26(05):1630013, 2016.
- [7] B. Gaudet, R. Linares, and R. Furfaro. Six degree-of-freedom hovering over an asteroid with unknown environmental dynamics via reinforcement learning. In *20th AIAA Scitech Forum*, Orlando, Florida, 2020.
- [8] B. Gaudet, R. Linares, and R. Furfaro. Terminal adaptive guidance via reinforcement meta-learning: Applications to autonomous asteroid close-proximity operations. *Acta Astronautica*, 171:1–13, 2020.
- [9] A. Rubinsztein, R. Sood, and F. E. Laipert. Neural network optimal control in astrodynamics: Application to the missed thrust problem. *Acta Astronautica*, 176:192–203, 2020.

- [10] T. A. Estlin, B. J. Bornstein, D. M. Gaines, R. C. Anderson, D. R. Thompson, M. Burl, R. Castaño, and M. Judd. Aegis automated science targeting for the mer opportunity rover. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3:1–19, 2012.
- [11] R. Francis, T. Estlin, G. Doran, S. Johnstone, D. Gaines, V. Verma, M. Burl, J. Frydenvang, S. Montano, R. Wiens, S. Schaffer, O. Gasnault, L. Deflores, D. Blaney, and B. Bornstein. Aegis autonomous targeting for chemcam on mars science laboratory: Deployment and results of initial science team use. *Science Robotics*, 2, 2017.
- [12] S. Higa, Y. Iwashita, K. Otsu, M. Ono, O. Lamarre, A. Didier, and M. Hoffmann. Vision-based estimation of driving energy for planetary rovers using deep learning and terramechanics. *IEEE Robotics and Automation Letters*, 4:3876–3883, 2019.
- [13] B. Rothrock, J. Papon, R. Kennedy, M. Ono, M. Heverly, and C. Cunningham. Spoc: Deep learning-based terrain classification for mars rover missions. In *AIAA Space and Astronautics Forum and Exposition, SPACE 2016*. American Institute of Aeronautics and Astronautics Inc, AIAA, 2016.
- [14] K. L. Wagstaff, G. Doran, A. Davies, S. Anwar, S. Chakraborty, M. Cameron, I. Daubar, and C. Phillips. Enabling onboard detection of events of scientific interest for the europa clipper spacecraft. In *25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2191–2201, Anchorage, Alaska, 2019.
- [15] B. Dachwald. Evolutionary neurocontrol: A smart method for global optimization of low-thrust trajectories. In *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, pages 1–16, Providence, Rhode Island, 2004.
- [16] S. D. Smet and D. J. Scheeres. Identifying heteroclinic connections using artificial neural networks. *Acta Astronautica*, 161:192–199, 2019.
- [17] N. L. O. Parrish. *Low Thrust Trajectory Optimization in Cislunar and Translunar Space*. PhD thesis, University of Colorado Boulder, 2018.
- [18] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. A. Riedmiller, and D. Silver. Emergence of locomotion behaviours in rich environments. *CoRR*, abs/1707.02286, 2017.
- [19] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den

- Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550, 2017.
- [20] R. Furfaro, A. Scorsoglio, R. Linares, and M. Massari. Adaptive generalized zemzhev feedback guidance for planetary landing via a deep reinforcement learning approach. *Acta Astronautica*, 171:156–171, 2020.
  - [21] B. Gaudet, R. Linares, and R. Furfaro. Deep reinforcement learning for six degrees of freedom planetary landing. *Advances in Space Research*, 65:1723–1741, 2020.
  - [22] B. Gaudet, R. Furfaro, and R. Linares. Reinforcement learning for angle-only intercept guidance of maneuvering targets. *Aerospace Science and Technology*, 99, 2020.
  - [23] D. Guzzetti. Reinforcement learning and topology of orbit manifolds for station-keeping of unstable symmetric periodic orbits. In *AAS/AIAA Astrodynamics Specialist Conference*, Portland, Maine, 2019.
  - [24] J. A. Reiter and D. B. Spencer. Augmenting spacecraft maneuver strategy optimization for detection avoidance with competitive coevolution. In *20th AIAA Scitech Forum*, Orlando, Florida, 2020.
  - [25] A. Das-Stuart, K. C. Howell, and D. C. Folta. Rapid trajectory design in complex environments enabled by reinforcement learning and graph search strategies. *Acta Astronautica*, 171:172–195, 2020.
  - [26] D. Miller and R. Linares. Low-thrust optimal control via reinforcement learning. In *29th AAS/AIAA Space Flight Mechanics Meeting*, Ka’anapali, Hawaii, 2019.
  - [27] C. J. Sullivan and N. Bosanac. Using reinforcement learning to design a low-thrust approach into a periodic orbit in a multi-body system. In *20th AIAA Scitech Forum*, Orlando, Florida, 2020.
  - [28] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015.
  - [29] J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 1889–1897, 2015.

- [30] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, pages 1928–1937, 2016. arXiv:1602.01783.
- [31] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. Continuous control with deep reinforcement learning, 2019.
- [32] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint*, arXiv:1707.06347, 2017.
- [33] S. Fujimoto, H. V. Hoof, and D. Meger. Addressing function approximation error in actor-critic methods. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1587–1596, 2018.
- [34] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1861–1870, 2018.
- [35] A. Kumar, A. Zhou, G. Tucker, and S. Levine. Conservative q-learning for offline reinforcement learning. In *Advances in Neural Information Processing Systems 33 (NeurIPS)*, pages 1179–1191, 2020.
- [36] K. Prudencio, J. L. Xiang, and A. T. Cemgil. A survey on offline reinforcement learning: Methodologies, challenges, and open problems. *arXiv preprint*, arXiv:2203.01387, 2022.
- [37] J. GarcÃa and F. FernÃ¡ndez. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(42):1437–1480, 2015.
- [38] F. Ghazalpour, S. Samangouei, and R. Vaughan. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys*, 54(12):1–35, 2021.
- [39] K. Song, J. Zhu, Y. Chow, D. Psomas, and M. Wainwright. A survey on multi-agent reinforcement learning: Foundations, advances, and open challenges. *IEEE Transactions on Neural Networks and Learning Systems*, 2024. In press, arXiv:2401.01234.
- [40] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. V. D. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach,

- K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- [41] O. Vinyals, I. Babuschkin, W. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.
- [42] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 1407–1416, 2018.
- [43] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the 10th International Conference on Machine Learning (ICML)*, pages 330–337, 1993.
- [44] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Robots*, 8(3):355–377, 2005.
- [45] L. Buşoniu, R. Babuška, and B. D. Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 38(2):156–172, 2008.
- [46] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*, pages 6379–6390, 2017.
- [47] P. Sunehag, G. Lever, A. Gruslys, W. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel. Value-decomposition networks for cooperative multi-agent learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2018. arXiv:1706.05296.
- [48] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 4292–4301, 2018.
- [49] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, J. Foerster, N. Nardelli, T. G. J. Rudner, and et al. The starcraft multi-agent challenge. *arXiv preprint*, arXiv:1902.04043, 2019.

- [50] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 5887–5896, 2019.
- [51] A. Mahajan, T. Rashid, M. Samvelyan, and S. Whiteson. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pages 7611–7622, 2019.
- [52] T. Wang, Y. Jiang, T. Da, W. Zhang, and J. Wang. Roma: Multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, pages 9876–9886, 2020.
- [53] K. Zhang, Z. Yang, and T. Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of RL and Control*, 2021. arXiv:2106.05230.
- [54] A. Mitriakov, P. Papadakis, J. Kerdreux, and S. Garlatti. Reinforcement learning based, staircase negotiation learning: Simulation and transfer to reality for articulated tracked robots. *IEEE Robotics & Automation Magazine*, 28(4):10–20, 2021.
- [55] Y. Yu et al. Heterogeneous-agent reinforcement learning: An overview. *IEEE Transactions on Neural Networks and Learning Systems*, 2022. In press, arXiv:2203.00596.
- [56] D. Vallado and W. McClain. *Fundamentals of Astrodynamics and Applications*. Fundamentals of Astrodynamics and Applications. Microcosm Press, 2001.
- [57] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. Pmlr, 2014.
- [58] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

- [59] S. Fujimoto, H. van Hoof, and D. Meger. Addressing function approximation error in actor-critic methods, 2018.
- [60] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. 2017.
- [61] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *CoRR*, abs/1801.01290, 2018.

## Abstract

In this study, a quadcopter stand with three degrees of freedom was controlled using game theory-based control. The first player tracks a desired input, and the second player creates a disturbance in the tracking of the first player to cause an error in the tracking. The move is chosen using the Nash equilibrium, which presupposes that the other player made the worst move.. In addition to being resistant to input interruptions, this method may also be resilient to modeling system uncertainty. This method evaluated the performance through simulation in the Simulink environment and implementation on a three-degree-of-freedom stand.

**Keywords:** Quadcopter, Differential Game, Game Theory, Nash Equilibrium, Three Degree of Freedom Stand, Model Base Design, Linear Quadratic Regulator



Sharif University of Technology  
Department of Aerospace Engineering

Master Thesis

# **Robust Reinforcement Learning Differential Game Guidance in Low-Thrust, Multi-Body Dynamical Environments**

By:

**Ali BaniAsad**

Supervisor:

**Dr.Hadi Nobahari**

December 2024