



دانشگاه صنعتی شریف  
دانشکده‌ی مهندسی هوافضا

پروژه کارشناسی  
مهندسی کنترل

عنوان:

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

نگارش:

علی بنی اسد

استاد راهنما:

دکتر هادی نوبهاری

تیر ۱۴۰۱





به نام خدا  
دانشگاه صنعتی شریف  
دانشکده‌ی مهندسی هوافضا

## پروژه کارشناسی

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

## کمیته‌ی ممتحنین

استاد راهنما: 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 بی‌دریغشان، 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 بی‌دریغش، نگاه مهربانش و دستان گرمش نبود برگ 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، مدل مبنا، تنظیم‌کننده مربعی خطی

---

<sup>۱</sup>Game Theory

<sup>۲</sup>Nash Equilibrium

# فهرست مطالب

۱	یادگیری تقویتی	۱
۱	۱-۱ مفاهیم اولیه	۱
۲	۱-۱-۱ حالت و مشاهدات	۲
۲	۱-۱-۲ فضای عمل	۲
۲	۱-۱-۳ سیاست	۲
۳	۱-۱-۴ مسیر	۳
۳	۱-۱-۵ تابع پاداش و بازگشت	۳
۴	۱-۱-۶ ارزش در یادگیری تقویتی	۴
۵	۲-۱ عامل گرادیان سیاست عمیق قطعی	۵
۵	۱-۲-۱ یادگیری Q در DDPG	۵
۷	۲-۲-۱ سیاست در DDPG	۷
۷	۳-۲-۱ اکتشاف و بهره‌برداری در DDPG	۷
۸	۴-۲-۱ شبکه‌کد	۸
۱۰	۳-۱ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه	۱۰
۱۰	۴-۱ بهینه‌سازی سیاست مجاور	۱۰
۱۱	۱-۴-۱ سیاست در الگوریتم PPO	۱۱
۱۱	۲-۴-۱ اکتشاف و بهره‌برداری در PPO	۱۱
۱۲	۳-۴-۱ شبکه‌کد PPO	۱۲

# فهرست جداول



# فهرست تصاویر

۱-۱ حلقه تعامل عامل و محیط	۲
----------------------------	---

# فهرست الگوریتم‌ها

۹	.....	گرایان سیاست عمیق قطعی	۱
۱۲	.....	بهینه‌سازی سیاست مجاور (PPO-Clip)	۲

# فصل ۱

## یادگیری تقویتی

### ۱-۱ مفاهیم اولیه

بخش‌های 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 تعامل دارد. 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 بر روی محیط عمل می‌کند، محیط تغییر می‌کند، اما ممکن است محیط به تنهایی نیز تغییر کند. عامل همچنین یک سیگنال پاداش<sup>۴</sup> از محیط دریافت می‌کند، عددی که به آن می‌گویند وضعیت فعلی محیط چقدر خوب یا بد است. هدف عامل به حداکثر رساندن پاداش انباشته خود است که بازگشت<sup>۵</sup> نام دارد. یادگیری تقویتی روش‌هایی هستند که عامل رفتارهای مناسب برای رسیدن به هدف خود را می‌آموزد. در شکل ۱-۱ تعامل بین محیط و عامل نشان داده شده است.

---

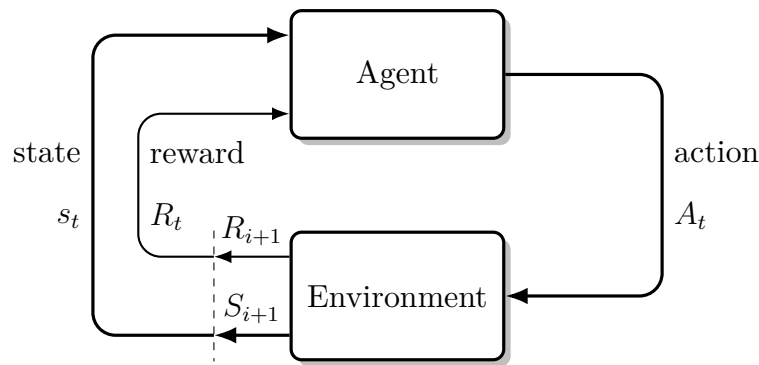
<sup>۱</sup> Reinforcement Learning (RL)

<sup>۲</sup> Agent

<sup>۳</sup> Environment

<sup>۴</sup> Reward

<sup>۵</sup> Return



شکل ۱-۱: حلقه تعامل عامل و محیط

### ۱-۱-۱ حالت و مشاهدات

حالت<sup>۶</sup> ( $s$ ) توصیف کاملی از وضعیت محیط است. همه‌ی اطلاعات محیط در حالت وجود دارد. مشاهده<sup>۷</sup> ( $o$ ) یک توصیف جزئی از حالت است که ممکن است شامل تمامی اطلاعات نباشد.

### ۲-۱-۱ فضای عمل

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

### ۳-۱-۱ سیاست

یک سیاست<sup>۱۰</sup> قاعده‌ای است که یک عامل برای تصمیم‌گیری در مورد اقدامات خود استفاده می‌کند. در این پژوهش سیاست قطعی<sup>۱۱</sup> است، که به صورت زیر نشان داده می‌شود:

$$a_t = \pi(s_t) \quad (1-1)$$

<sup>6</sup>State

<sup>7</sup>Observation

<sup>8</sup>discrete

<sup>9</sup>continuous

<sup>10</sup>policy

<sup>11</sup>deterministic

در یادگیری تقویتی عمیق از سیاست‌های پارامتری شده استفاده می‌شود. خروجی این سیاست‌ها از توابعی هستند که به مجموعه‌ای از پارامترها (مثلاً وزن‌ها و بایاس‌های یک شبکه عصبی) بستگی دارند که می‌توان آنها را برای تغییر رفتار از طریق برخی الگوریتم‌های بهینه‌سازی تنظیم کرد. در این پژوهش پارامترهای سیاست را با  $\theta$  نشان داده شده است و سپس نماد آن به عنوان یک زیروند روی سیاست مانند معادله (۲-۱) نشان داده شده است.

$$a_t = \pi_\theta(s_t) \quad (2-1)$$

## ۴-۱-۱ مسیر

یک مسیر<sup>۱۲</sup> توالی‌ای از حالت‌ها و عمل‌ها در محیط است.

$$\tau = (s_0, a_0, s_1, a_1, \dots) \quad (3-1)$$

گذار حالت<sup>۱۳</sup> به اتفاقاتی که در محیط بین حالت در زمان  $s$  و حالت در زمان  $s+1$  می‌افتد، گفته می‌شود. این گذارها توسط قوانین طبیعی محیط انجام می‌شوند و تنها به آخرین اقدام انجام شده توسط عامل ( $a_t$ ) بستگی دارند. گذار حالت را می‌توان به صورت زیر تعریف کرد.

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

## ۵-۱-۱ تابع پاداش و بازگشت

تابع پاداش<sup>۱۴</sup> حالت فعلی محیط، آخرین عمل انجام شده و حالت بعدی محیط بستگی دارد. تابع پاداش را می‌توان به صورت زیر تعریف کرد.

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

در این پژوهش پاداش تنها تابعی از جفت حالت-عمل ( $r_t = R(s_t, a_t)$ ) است. هدف عامل این است که مجموع پاداش‌های به دست آمده در طول یک مسیر را به حداکثر برساند، اما این مفهوم می‌تواند چند معنی داشته باشد. در این پژوهش این موارد را با نماد  $R(\tau)$  نشان داده شده است و به آن تابع بازگشت<sup>۱۵</sup> گفته می‌شود. یکی از انواع بازگشت، بازگشت بدون تنزیل با افق محدود<sup>۱۶</sup> است که مجموع پاداش‌های به دست آمده در یک

<sup>12</sup>Trajectory

<sup>13</sup>state transition

<sup>14</sup>reward function

<sup>15</sup>Return

<sup>16</sup>Finite-Horizon Undiscounted Return

بازه زمانی ثابت از مسیر به صورت زیر است.

$$R(\tau) = \sum_{t=0}^T r_t \quad (6-1)$$

نوع دیگری از بازگشت، بازگشت تنزیل شده با افق نامحدود<sup>۱۷</sup> است که مجموع همه پاداش‌هایی است که تا به حال توسط عامل به دست آمده است، اما با در نظر گرفتن فاصله زمانی‌ای که تا دریافت آن پاداش وجود داشته، تنزیل<sup>۱۸</sup> شده است. این فرمول پاداش شامل یک فاکتور تنزیل<sup>۱۹</sup> با نماد  $\gamma$  است.

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \quad (7-1)$$

## ۶-۱-۱ ارزش در یادگیری تقویتی

در یادگیری تقویتی، دانستن ارزش<sup>۲۰</sup> یک حالت یا جفت حالت-عمل ضروری است. منظور از ارزش، بازگشت مورد انتظار<sup>۲۱</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 کرد. توابع ارزش به شکلی در تقریباً تمام الگوریتم‌های یادگیری تقویتی به کار می‌روند. در اینجا به چهار تابع مهم اشاره می‌کنیم.

۱. تابع ارزش تحت سیاست<sup>۲۲</sup>  $(V^\pi(s))$ : این تابع، بازگشت مورد انتظار را در صورتی که از حالت  $s$  شروع شود و همیشه طبق سیاست  $\pi$  عمل شود، خروجی می‌دهد.

$$V^\pi(s) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s] \quad (8-1)$$

۲. تابع ارزش-عمل تحت سیاست<sup>۲۳</sup>  $(Q^\pi(s, a))$ : این تابع، بازگشت مورد انتظار را در صورتی که از حالت  $s$  شروع شود، یک اقدام دلخواه  $a$  (که ممکن است از سیاست  $\pi$  نباشد) انجام شود و سپس برای همیشه طبق سیاست  $\pi$  عمل شود، خروجی می‌دهد.

$$Q^\pi(s, a) = \mathbb{E}_{\tau \sim \pi} [R(\tau) | s_0 = s, a_0 = a] \quad (9-1)$$

۳. تابع ارزش بهینه<sup>۲۴</sup>  $(V^*(s))$ : این تابع، بازگشت مورد انتظار را در صورتی که از حالت  $s$  شروع شود و همیشه طبق سیاست بهینه در محیط عمل شود، خروجی می‌دهد.

<sup>17</sup>Infinite-Horizon Discounted Return

<sup>18</sup>Discount

<sup>19</sup>Discount Factor

<sup>20</sup>Value

<sup>21</sup>Expected Return

<sup>22</sup>On-Policy Value Function

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

<sup>24</sup>Optimal Value Function

$$V^*(s) = \max_{\pi}(V^{\pi}(s)) \quad (۱۰-۱)$$

۴. تابع ارزش-عمل بهینه<sup>۲۵</sup> ( $Q^*(s, a)$ ): این تابع، بازگشت مورد انتظار را 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، خروجی می‌دهد.

$$Q^*(s, a) = \max_{\pi}(Q^{\pi}(s, a)) \quad (۱۱-۱)$$

## ۲-۱ عامل گرادیان سیاست عمیق قطعی

گرادیان سیاست عمیق قطعی<sup>۲۶</sup> الگوریتمی است که همزمان یک تابع  $Q$  و یک سیاست را یاد می‌گیرد. این الگوریتم برای یادگیری تابع  $Q$  از داده‌های غیرسیاست محور<sup>۲۷</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 داده شده عمل بهینه را می‌توان با حل کردن معادله (۱۲-۱) به دست آورد.

$$a^*(s) = \arg \max_a Q^*(s, a) \quad (۱۲-۱)$$

الگوریتم DDPG ترکیبی از یادگیری تقریبی برای  $Q^*(s, a)$  و یادگیری تقریبی برای  $a^*(s)$  است و به نحوی طراحی شده است که برای محیط‌هایی با فضاهاى عمل پیوسته مناسب باشد. روش محاسبه  $a^*(s)$  در این الگوریتم آن را برای فضای پیوسته مناسب می‌کند. از آنجا که فضای عمل پیوسته است، فرض می‌شود که تابع  $Q^*(s, a)$  نسبت به آرگومان عمل مشتق‌پذیر است. مشتق‌پذیری این امکان را می‌دهد که یک روش یادگیری مبتنی بر گرادیان برای سیاست  $\mu(s)$  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.

### ۱-۲-۱ یادگیری $Q$ در DDPG

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

$$Q^*(s, a) = \mathbb{E}_{s' \sim P} \left[ r(s, a) + \gamma \max_{a'} Q^*(s', a') \right] \quad (۱۳-۱)$$

<sup>25</sup>Optimal Action-Value Function

<sup>26</sup>Deep Deterministic Policy Gradient (DDPG)

<sup>27</sup>Off-Policy

جمله  $s' \sim P$  به این معنی است که وضعیت بعدی  $s'$  توسط محیط از توزیع احتمال  $P(\cdot|s, a)$  نمونه‌گرفته می‌شود. معادله بلمن نقطه شروع برای یادگیری  $Q^*(s, a)$  با یک مقداردهی تقریبی است. پارامترهای یک شبکه عصبی  $Q_\phi(s, a)$  با علامت  $\phi$  نشان داده شده‌است. یک مجموعه  $\mathcal{D}$  از تغییر از یک حالت به حالت دیگر  $(s, a, r, s', d)$  (که  $d$  نشان می‌دهد که آیا وضعیت  $s'$  پایانی است یا خیر) جمع‌آوری شده‌است. یک تابع خطای میانگین مربعات بلمن (MSBE) استفاده شده‌است که معیاری برای نزدیکی  $Q_\phi$  برای برآورده کردن معادله بلمن است.

$$L(\phi, \mathcal{D}) = \mathbb{E}_{(s, a, r, s', d) \sim \mathcal{D}} \left[ \left( Q_\phi(s, a) - \left( r + \gamma(1 - d) \max_{a'} Q_\phi(s', a') \right) \right)^2 \right] \quad (۱۴-۱)$$

در الگوریتم DDPG دو ترفند برای عملکرد بهتر استفاده شده‌است که در ادامه به بررسی آن پرداخته شده‌است.

#### • بافرهای بازی

الگوریتم‌های یادگیری تقویتی جهت آموزش یک شبکه عصبی عمیق برای تقریب  $Q^*(s, a)$  از بافرهای بازی<sup>۲۸</sup> تجربه‌شده استفاده می‌کنند. این مجموعه  $\mathcal{D}$  شامل تجربیات قبلی است. برای داشتن رفتار پایدار در الگوریتم، بافر بازی باید به اندازه کافی بزرگ باشد تا شامل یک دامنه گسترده از تجربیات شود. انتخاب داده‌های بافر به دقت انجام شده‌است چرا که اگر فقط از داده‌های بسیار جدید استفاده شود، بیش‌برازش<sup>۲۹</sup> رخ می‌دهد 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.

$$r + \gamma(1 - d) \max_{a'} Q_\phi(s', a') \quad (۱۵-۱)$$

در هنگام کمینه کردن تابع خطای میانگین مربعات بلمن، سعی شده‌است تا تابع  $Q$  شبیه‌تر به این هدف یعنی رابطه (۱۵-۱) 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 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 دنباله‌ی شبکه اول را دنبال می‌کند. پارامترهای شبکه هدف با نشان  $\phi_{\text{targ}}$  نشان داده می‌شوند. در الگوریتم DDPG، شبکه هدف در هر

<sup>28</sup>Replay Buffers

<sup>29</sup>Overfit



به روزرسانی شبکه اصلی، با میانگین‌گیری پولیاک<sup>۳۰</sup> به روزرسانی می‌شود.

$$\phi_{\text{targ}} \leftarrow \rho \phi_{\text{targ}} + (1 - \rho) \phi \quad (۱۶-۱)$$

در رابطه بالا  $\rho$  یک فرایارامتر<sup>۳۱</sup> است که بین صفر و یک انتخاب می‌شود. در این پژوهش این مقدار نزدیک به یک در نظر گرفته شده است.

الگوریتم DDPG نیاز به یک شبکه سیاست هدف ( $\mu_{\theta_{\text{targ}}}$ ) برای محاسبه عمل‌هایی که به طور تقریبی بیشینه  $Q_{\phi_{\text{targ}}}$  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 در نظر گرفتن موارد اشاره شده، یادگیری  $Q$  در DDPG با کمینه کردن تابع خطای میانگین مربعات بلمن (MSBE) یعنی معادله (۱۷-۱) با استفاده از کاهش گرادیان تصادفی<sup>۳۲</sup> انجام می‌شود.

$$L(\phi, \mathcal{D}) = \mathbb{E}_{(s,a,r,s',d) \sim \mathcal{D}} \left[ \left( Q_{\phi}(s,a) - (r + \gamma(1-d)Q_{\phi_{\text{targ}}}(s', \mu_{\theta_{\text{targ}}}(s'))) \right)^2 \right] \quad (۱۷-۱)$$

## ۲-۲-۱ سیاست در DDPG

در این بخش یک سیاست تعیین شده  $\mu_{\theta}(s)$  یاد گرفته 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 مشتق پذیر است، معادله زیر با استفاده از صعود گرادیان<sup>۳۳</sup> (تنها نسبت به پارامترهای سیاست) حل می‌شود.

$$\max_{\theta} \mathbb{E}_{s \sim \mathcal{D}} [Q_{\phi}(s, \mu_{\theta}(s))] \quad (۱۸-۱)$$

## ۳-۲-۱ اکتشاف و بهره‌برداری در DDPG

برای بهبود اکتشاف<sup>۳۴</sup> در سیاست‌های DDPG، در زمان آموزش نویز به عمل‌ها اضافه می‌شود. نویسندگان مقاله اصلی DDPG توصیه کرده‌اند که نویز OU<sup>۳۵</sup> با زمان‌بندی هم‌ارتباطی<sup>۳۶</sup> اضافه شود. در زمان سنجش بهره‌برداری<sup>۳۷</sup> سیاست از آنچه یاد گرفته است، نویز به عمل‌ها اضافه نمی‌شود.

<sup>30</sup>Polyak Averaging

<sup>31</sup>Hyperparameter

<sup>32</sup>Stochastic Gradient Descent

<sup>33</sup>Gradient Ascent

<sup>34</sup>Exploration

<sup>35</sup>Ornstein-Uhlenbeck

<sup>36</sup>Time-Correlated

<sup>37</sup>Exploitation

## ۴-۲-۱ شبکه‌کد DDPG

در این بخش الگوریتم DDPG پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم ۱ در محیط پایتون با استفاده از کتابخانه TensorFlow [۱] پیاده‌سازی شده است.

---

## الگوریتم ۱ گرادیان سیاست عمیق قطعی

---

ورودی: پارامترهای اولیه سیاست  $(\theta)$ ، پارامترهای تابع  $Q(\phi)$ ، بافر بازی خالی  $(D)$

۱: پارامترهای هدف را برابر با پارامترهای اصلی قرار دهید  $\phi_{\text{targ}} \leftarrow \phi, \theta_{\text{targ}} \leftarrow \theta$

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  $\epsilon \sim \mathcal{N}$  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 به روزرسانی فرا رسیده است:

۸: به ازای هر تعداد به روزرسانی:

۹: یک دسته تصادفی گذر از یک حالت به حالت دیگر،  $B = \{(s, a, r, s', d)\}$ ، از  $D$  نمونه‌گیری شود.

۱۰: اهداف را محاسبه کنید:

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

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

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

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

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

۱۳: شبکه‌های هدف را با استفاده از معادلات زیر به روزرسانی کنید:

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

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

---

## ۳-۱ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه

## ۴-۱ بهینه‌سازی سیاست مجاور

الگوریتم بهینه‌سازی سیاست مجاور<sup>۳۸</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>۳۹</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، به دنبال 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 دارای 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 به دنبال حل تقریبی و به‌روزرسانی با محدودیت واگرایی کولباک-لیبلر<sup>۴۰</sup> است، مشابه روشی که در الگوریتم TRPO استفاده شده است. با این حال، به جای اعمال یک محدودیت سخت<sup>۴۱</sup>، PPO-Penalty واگرایی KL را در تابع هدف جریمه می‌کند. این جریمه 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، PPO-Clip از یک عملیات بریدن<sup>۴۲</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>38</sup>Proximal Policy Optimization (PPO)

<sup>39</sup>Trust Region Policy Optimization

<sup>40</sup>Kullback-Leibler (KL) Divergence

<sup>41</sup>Hard Constraint

<sup>42</sup>Clipping

## ۱-۴-۱ سیاست در الگوریتم PPO

تابع سیاست در الگوریتم PPO به صورت یک شبکه عصبی پیچیده 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  $\tanh$  استفاده شده است. تابع سیاست در الگوریتم PPO به صورت زیر به‌روزرسانی می‌شود:

$$\theta_{k+1} = \arg \max_{\theta} \mathbb{E}_{s,a \sim \pi_{\theta_k}} [L(s, a, \theta_k, \theta)] \quad (۱۹-۱)$$

در این پژوهش برای به حداکثر رساندن تابع هدف، چندین گام بهینه‌سازی گرادیان کاهشی تصادفی<sup>۴۳</sup> اجرا شده است. در معادله بالا  $L$  به صورت زیر تعریف شده است:

$$L(s, a, \theta_k, \theta) = \min \left( \frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)} A^{\pi_{\theta_k}}(s, a), \text{clip} \left( \frac{\pi_{\theta}(a|s)}{\pi_{\theta_k}(a|s)}, 1 - \epsilon, 1 + \epsilon \right) A^{\pi_{\theta_k}}(s, a) \right) \quad (۲۰-۱)$$

که در آن  $\epsilon$  یک فرایمتر است که مقدار آن معمولاً کوچک است. این فرایمتر مشخص می‌کند که چقدر اندازه گام بهینه‌سازی باید محدود شود. در این پژوهش مقدار  $\epsilon = 0.2$  انتخاب شده است.

در حالی که این نوع محدود کردن (PPO-Clip) تا حد زیادی به اطمینان از به‌روزرسانی‌های معقول سیاست کمک می‌کند، همچنان ممکن است با سیاست به‌دست 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>۴۴</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.

## ۲-۴-۱ اکتشاف و بهره‌برداری در PPO

الگوریتم PPO از یک سیاست تصادفی به صورت سیاست محور 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

<sup>۴۳</sup>Stochastic Gradient Descent (SGD)

<sup>۴۴</sup>Early Stopping

می‌کند تا از پاداش‌هایی که قبلاً پیدا کرده است، بهره‌برداری کند. البته این موضوع می‌تواند منجر به گیر افتادن خط‌مشی در بهینه‌های محلی<sup>۴۵</sup> شود.

### ۳-۴-۱ شبه‌کد PPO

در این بخش الگوریتم PPO پیاده‌سازی شده آورده شده است. در این پژوهش الگوریتم ۲ در محیط پایتون با استفاده از کتابخانه PyTorch [۲] پیاده‌سازی شده است.

---

#### الگوریتم ۲ بهینه‌سازی سیاست مجاور (PPO-Clip)

---

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

۱: به ازای  $k = 0, 1, 2, \dots$ :

۲: مجموعه‌ای از مسیرها به نام  $\mathcal{D}_k = \{\tau_i\}$  با اجرای سیاست  $\pi_k = \pi(\theta_k)$  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

فعلی  $V_{\phi_k}$

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

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

معمولاً از طریق گرادیان افزایشی تصادفی Adam.

۶: برازش تابع ارزش با رگرسیون بر روی میانگین مربعات خطا:

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

معمولاً از طریق برخی از الگوریتم‌های کاهش‌گرادیان.

---

<sup>45</sup>Local Optima

# Bibliography

- [1] 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.
- [2] 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.

## **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

Bachelor Thesis

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

By:

**Ali BaniAsad**

Supervisor:

**Dr.Hadi Nobahari**

July 2022