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

بسم الله الرحمن الرحیم

**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[1] به‌منظور کنترل وضعیت استند سه درجه آزادی چهارپره استفاده شده‌است. 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[2] 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 شده‌است و سپس، با پیاده‌سازی روی استند سه درجه آزادی صحت عملکرد کنترل‌کننده تایید شده‌است.



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


[1] Game Theory
[2] Nash Equilibrium

# فهرست مطالب

# فهرست جدول‌ها

# فهرست شکل‌ها

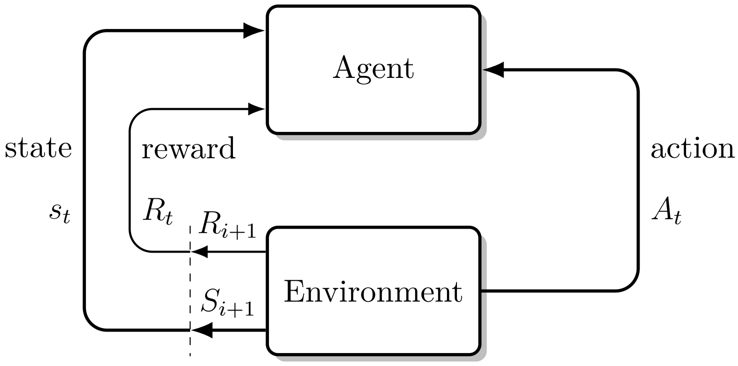
# فصل ۱

# یادگیری تقویتی

## ۱ـ۱ مفاهیم اولیه

بخش‌های اصلی یادگیری تقویتی[۱] شامل عامل[۲] و محیط[۳] 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 می کند، محیط تغییر می‌کند، اما ممکن است محیط به تنهایی نیز تغییر کند. عامل همچنین یک سیگنال پاداش[۴] از محیط دریافت می کند، عددی که به آن می‌گوید وضعیت فعلی محیط چقدر خوب یا بد است. هدف عامل به حداکثر رساندن پاداش انباشته خود است که بازگشت[۵] نام دارد. یادگیری تقویتی روش‌هایی هستند که عامل رفتارهای مناسب برای رسیدن به هدف خود را می‌آموزد. در شکل ۱ـ۱ تعامل بین محیط و عامل نشان داده شده است.

[۱] Reinforcement Learning (RL)
[۲] Agent
[۳] Environment
[۴] Reward
[۵] Return




**شکل ۱ـ۱: حلقه تعامل عامل و محیط**

### ۱ـ۱ـ۱ حالت و مشاهدات

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

### ۱ـ۱ـ۲ فضای عمل

فضای عمل در یادگیری تقویتی، مجموعه‌ای از تمام اقدامات است که یک عامل می‌تواند در محیط خود انجام دهد. این فضا می‌تواند گسسته[۸] یا پیوسته[۹] باشد. در این پژوهش فضای عمل پیوسته و در یک بازه مشخص است.

### ۱ـ۱ـ۳ سیاست

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

$$a_t = \pi(s_t) \qquad (۱ـ۱)$$

[۶] State
[۷] Observation
[۸] discrete
[۹] continuous
[۱۰] policy
[۱۱] deterministic

در یادگیری تقویتی عمیق از سیاست‌های پارامتری‌شده 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 (۱ـ۲) نشان داده شده است.

$$a_t = \pi_\theta(s_t) \tag{۱ـ۲}$$

### ۱ـ۱ـ۴ مسیر

یک مسیر[۱۲] توالی‌ای از حالت‌ها و عمل‌ها در محیط است.

$$\tau = (s_0, a_0, s_1, a_1, \cdots) \tag{۱ـ۳}$$

گذار حالت[۱۳] 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 به‌صورت زیر تعریف کرد.

$$s_{t+1} = f(s_t, a_t) \tag{۱ـ۴}$$

### ۱ـ۱ـ۵ تابع پاداش و بازگشت

تابع پاداش[۱۴] حالت فعلی محیط، آخرین عمل انجام شده و حالت بعدی محیط بستگی دارد. تابع پاداش را می‌توان به‌صورت زیر تعریف کرد.

$$r_t = R(s_t, a_t, s_{t+1}) \tag{۱ـ۵}$$

در این پژوهش پاداش تنها تابعی از جفت حالت‌ـعمل $(r_t = R(s_t, a_t))$ است. هدف عامل این است که مجموع پاداش‌های به‌دست‌آمده 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[۱۵]

---

[۱۲] Trajectory
[۱۳] state transition
[۱۴] reward function
[۱۵] Return

گفته می‌شود. یکی از انواع بازگشت، بازگشت بدون تنزیل با افق محدود[16] است که مجموع پاداش‌های به‌دست‌آمده در یک بازه زمانی ثابت از مسیر به‌صورت زیر است.

$$R(\tau) = \sum_{t=0}^{T} r_t \tag{۱ـ۶}$$

نوع دیگری از بازگشت، بازگشت تنزیل‌شده با افق نامحدود[17] 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[18] شده است. این فرمول پاداش شامل یک فاکتور تنزیل[19] با نماد $\gamma$ است.

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r_t \tag{۱ـ۷}$$

## ۱ـ۱ـ۶ ارزش در یادگیری تقویتی

در یادگیری تقویتی، دانستن ارزش[20] یک حالت یا جفت حالت‌ـعمل ضروری است. منظور از ارزش، بازگشت مورد انتظار[21] است، یعنی اگر از آن حالت یا جفت حالت‌ـعمل 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 تقریبا تمام الگوریتم‌های یادگیری تقویتی به کار می‌روند. در اینجا به چهار تابع مهم اشاره می‌کنیم.

۱. تابع ارزش تحت سیاست[22] ($V^\pi(s)$): این تابع، بازگشت مورد انتظار را در صورتی که از حالت $s$ شروع شود و همیشه طبق سیاست $\pi$ عمل شود، خروجی می‌دهد.

$$V^\pi(s) = \underset{\tau\sim\pi}{\mathbb{E}}[R(\tau)|s_0 = s] \tag{۱ـ۸}$$

۲. تابع ارزش‌ـعمل تحت سیاست[23] ($Q^\pi(s,a)$): 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.

$$Q^\pi(s,a) = \underset{\tau\sim\pi}{\mathbb{E}}[R(\tau)|s_0 = s, a_0 = a] \tag{۱ـ۹}$$

[16] Finite-Horizon Undiscounted Return
[17] Infinite-Horizon Discounted Return
[18] Discount
[19] Discount Factor
[20] Value
[21] Expected Return
[22] On-Policy Value Function
[23] On-Policy Action-Value Function

۳. تابع ارزش بهینه[۲۴] ($V^*(s)$): 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.

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

۴. تابع ارزش-عمل بهینه[۲۵] ($Q^*(s,a)$): 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.

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

## ۱-۲ عامل گرادیان سیاست عمیق قطعی

گرادیان سیاست عمیق قطعی[۲۶] الگوریتمی است که همزمان یک تابع Q و یک سیاست را یاد می‌گیرد. این الگوریتم برای یادگیری تابع Q از داده‌های غیرسیاست محور[۲۷] 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 (۱-۱۲) به دست آورد.

$$a^*(s) = \arg\max_{a} Q^*(s,a) \tag{۱۲-۱}$$

الگوریتم DDPG ترکیبی از یادگیری تقریبی برای $Q^*(s,a)$ و یادگیری تقریبی برای $a^*(s)$ است و به نحوی طراحی شده‌است که برای محیط‌هایی با فضاهای 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.

[۲۴] Optimal Value Function
[۲۵] Optimal Action-Value Function
[۲۶] Deep Deterministic Policy Gradient (DDPG)
[۲۷] Off-Policy

### ۱ـ۲ـ۱ یادگیری Q در DDPG

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

$$Q^*(s,a) = \mathop{\mathrm{E}}_{s' \sim P}\left[r(s,a) + \gamma \max_{a'} Q^*(s',a')\right] \tag{۱ـ۱۳}$$

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.

$$L(\phi,\mathcal{D}) = \mathop{\mathrm{E}}_{(s,a,r,s',d)\sim\mathcal{D}}\left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d)\max_{a'} Q_\phi(s',a')\right)\right)^2\right] \tag{۱ـ۱۴}$$

در الگوریتم DDPG دو ترفند برای عمکرد 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[۲۸] 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[۲۹] 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 به‌عنوان هدف شناخته می‌شود.

$$r + \gamma(1-d)\max_{a'} Q_\phi(s',a') \tag{۱ـ۱۵}$$

[۲۸] Replay Buffers
[۲۹] Overfit

در هنگام کمینه کردن تابع خطای میانگین مربعات بلمن، سعی شده‌است تا تابع Q شبیه‌تر به این هدف یعنی رابطه (۱ـ۱۵) 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 به‌روزرسانی شبکه اصلی، با میانگین‌گیری پولیاک[۳۰] به‌روزرسانی می‌شود.

$$\phi_{\text{targ}} \leftarrow \rho\phi_{\text{targ}} + (1-\rho)\phi \qquad (۱ـ۱۶)$$

در رابطه بالا $\phi$ یک فراپارامتر[۳۱] است که بین صفر و یک انتخاب می‌شود. در این پژوهش این مقدار نزدیک به یک درنظرگرفته شده‌است.

الگوریتم DDPG نیاز به یک شبکه سیاست هدف ($\mu_{\theta_{\text{targ}}}$) برای محاسبه عمل‌هایی که به‌طور 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 درنظرگرفتن موارد اشاره‌شده، یادگیری Q در DDPG با کمینه کردن تابع خطای میانگین مربعات بلمن (MSBE) یعنی معادله (۱ـ۱۷) با استفاده از کاهش گرادیان تصادفی[۳۲] انجام می‌شود.

$$L(\phi,\mathcal{D}) = \underset{(s,a,r,s',d)\sim\mathcal{D}}{\mathrm{E}}\left[\left(Q_\phi(s,a) - \left(r + \gamma(1-d)Q_{\phi_{\text{targ}}}(s',\mu_{\theta_{\text{targ}}}(s'))\right)\right)^2\right] \qquad (۱ـ۱۷)$$

## ۱ـ۲ـ۲ سیاست در DDPG

در این بخش یک سیاست تعیین‌شده $\mu_\theta(s)$ یادگرفته می‌شود تا عملی را انجام می‌دهد که بیشینه $Q_\phi(s,a)$ رخ دهد. از آنجا که فضای عمل پیوسته است و فرض شده‌است که تابع Q نسبت به عمل مشتق‌پذیر است، معادله زیر با استفاده از صعود گرادیان[۳۳] (تنها نسبت به پارامترهای سیاست) حل می‌شود.

$$\max_\theta \underset{s\sim\mathcal{D}}{\mathrm{E}}\left[Q_\phi(s,\mu_\theta(s))\right] \qquad (۱ـ۱۸)$$

[۳۰] Polyak Averaging
[۳۱] Hyperparameter
[۳۲] Stochastic Gradient Descent
[۳۳] Gradient Ascent

### ۱ـ۲ـ۳ اکتشاف و بهره‌برداری در DDPG

برای بهبود اکتشاف[۳۴] در سیاست‌های DDPG، در زمان آموزش نویز به عمل‌ها اضافه می‌شود. نویسندگان مقاله اصلی DDPG توصیه کرده‌اند که نویز OU[۳۵] با زمان‌بندی هم‌ارتباطی[۳۶] اضافه شود، اما نتایج به‌روزتر نشان می‌دهد که نویز گوسی بدون هم‌ارتباط[۳۷] و میانگین صفر کاملاً موثر عمل می‌کند. از آنجا که نویز گوسی با میانگین صفر ساده‌تر است، در این پژوهش از این روش استفاده شده‌است. در زمان سنجش بهره‌برداری[۳۸] سیاست از آنچه یادگرفته است، نویز به عمل‌ها اضافه نمی‌شود.

### ۱ـ۲ـ۴ شبه‌کد

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

---

[۳۴] Exploration
[۳۵] Ornstein–Uhlenbeck
[۳۶] Time-Correlated
[۳۷] Uncorrelated
[۳۸] Exploitation

**الگوریتم ۱** گرادیان سیاست عمیق قطعی

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

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

تا وقتی همگرایی رخ دهد:

وضعیت ($s$) را مشاهده کرده و عمل $a = \text{clip}(\mu_\theta(s) + \epsilon, a_{\text{Low}}, a_{\text{High}})$ را انتخاب کنید، به‌طوری که $\epsilon \sim \mathcal{N}$ است.

عمل $a$ را در محیط اجرا کنید.

وضعیت بعدی $s'$، پاداش $r$ و سیگنال پایان $d$ را مشاهده کنید تا نشان دهد آیا $s'$ پایانی است یا خیر.

اگر $s'$ پایانی است، وضعیت محیط را بازنشانی کنید.

اگر زمان به‌روزرسانی فرا رسیده است:

به ازای هر تعداد به‌روزرسانی:

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

اهداف 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:

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

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

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

شبکه‌های هدف را با استفاده از معادلات زیر به‌روزرسانی کنید:

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

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

## ۱ـ۳ عامل گرادیان سیاست عمیق قطعی تاخیری دوگانه

# مراجع

**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