به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

بهینه سازی و یادگیری توزیع شده

پروژه 1

نام و نام خانوادگی	نیما زمان پور
شماره دانشجویی	810198407
تاریخ ارسال گزارش	دی 1402

فهرست

شکل‌ها

modeling

در این پروژه قصد داریم به کمک روش **Model Predictive Control(mpc)** یک مسئله **path planning** را حل کنیم. در این مسئله ورودی سرعت در راستای **x,y** است و عامل ها باید با حفظ یک چینش قابل تنظیم در طول مسیر از تعدادی نقطه اجباری عبور کنند. و بقیه نقاط را بیابند. نقاط اجباری فقط به عامل اول داده می‌شود. همچنین نیروی ورودی باید بهینه باشد. پس داریم:

$$horizon: length\ of\ route$$

$$X_i = \left[x_i, y_i, V_{x_i}, V_{y_i}\right] \rightarrow (horizon * 4)$$

$$f(X) = \sum_{i=1}^{n_{agent}} f_{path_i}(X_i) + f_{cost_i}(X_i)$$

$$s.t.: x_{i,j+1} = x_{i,j} + V_{x(i,j+1)}dt$$
$$y_{i,j+1} = y_{i,j} + V_{y(i,j+1)}dt$$
$$\forall j \in Horizon, \forall i \in n_{agent} - \{1\} : x_{i,j} = x_{1,j} + keepDist[i][j][0]$$
$$y_{i,j} = y_{1,j} + keepDist[i][j][1]$$

$$f_{path}(X) = \sum 10 * \left(X\left[index_{checkpoint}\right] - checkpoints\right)^2$$

$$f_{cost}(X) = \left(\|V_x\|^2 + \left\|V_y\right\|^2\right)/10$$

برای طبیعی تر کردن چرخش عامل ها، فاصله بین عامل ها و عامل اول یکسان نیست. بلکه با جهت سرعت عامل اول تغییر می‌کند. این تغییر بصورت یک ماتریس دوران در $keepDist[i]$ ضرب می‌شود.

$$\forall i \in Horizon : \theta_i = arctan\left(\frac{V_{y,i}}{V_{x,i}}\right)$$

$$keepDist[i][j] = initialKeepDist[i] * rotationMatrix(\theta - \frac{\pi}{2})$$

Dual Decomposition

الف) توضیحات

در این روش هر عامل horizon*4 متغیر، horizon*2 قید داخلی(مربوط به معادله سرعت و horizon*2 قید مشترک(مربوط به حفظ فاصله) دارد.

قید داخلی را $D(x) = 0$ و قید خارجی را $K(x)=0$ نام گذاری می‌کنیم.

حال لاگرانژین را می‌نویسیم:

$$L(X,\mu,\lambda) = \mathrm{f(X)} + \mathrm{\mu D(X)} + \mathrm{\lambda K(X)}$$

سپس صورت تجزیه شده آن را می‌نویسیم.

$$L_i(X_i,\mu_i,\lambda_i) = \mathrm{f}_i(\mathrm{X_i}) + \mathrm{\mu_i D}_i(\mathrm{X_i}) + \lambda_i * \begin{cases} i = 1 & -X_{x,y,i} * (i-1) \\ i \neq 1 & X_{x,y,i} - KeepDist[i-1] \end{cases}$$

بعد از تجزیه شدن لاگرانژین عامل ها، می‌توان مسئله را بصورت دوسطحی حل کرد بطوری که

1. ابتدا مقادیر X,μ,λ را بصورت رندوم مقدار دهی اولیه می‌کنیم.
2. سپس هر عامل در سطح پایین یک بهینه سازی برای تابع لاگرانژین خود روی X,μ انجام می‌دهد.
   a. For 100 times do:
   b. For all agents do:
   c. $X_i^{k+1} = X_i^k - \alpha * \frac{\partial}{\partial X_i} L_i(X,\mu^k,\lambda^l)$
   d. $\mu_i^{k+1} = \mu_i^k + \alpha * \frac{\partial}{\partial \mu_i} L_i(X^k,\mu,\lambda^l)$
3. در سطح بالا ضرایب قید های مشترک آپدیت می‌شوند.
   a. $\lambda_i^{l+1} = \lambda_i^k + \alpha * \frac{\partial}{\partial \lambda_i} L_i(X^k,\mu^k,\lambda)$

ب) شبیه سازی

مقادیر اولیه شبیه سازی:

$$N = 3, dt = 1$$

$$[x_0, y_0] = \left[\begin{bmatrix}0\\1\end{bmatrix} \quad \begin{bmatrix}-1\\0\end{bmatrix} \quad \begin{bmatrix}1\\0\end{bmatrix}\right]$$

$$checkpoints = \left[\begin{bmatrix}0\\20\end{bmatrix} \quad \begin{bmatrix}20\\20\end{bmatrix} \quad \begin{bmatrix}20\\0\end{bmatrix}\right]$$

نرخ یادگیری lr = 0.05

در این حالت سطح داخلی 100 مرتبه و سطح بیرونی 1000 مرتبه آپدیت شدند.

نتایج بصورت زیر است:

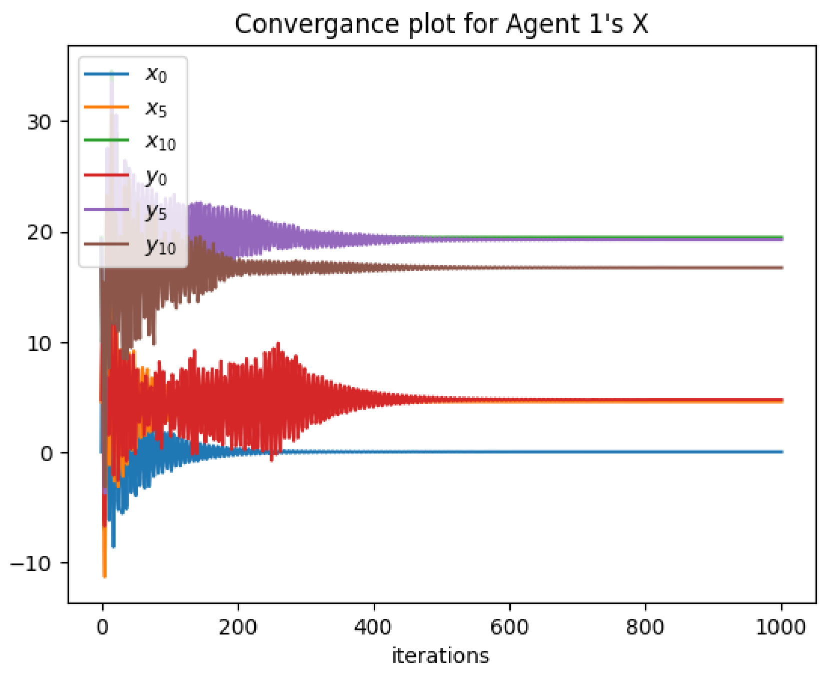

Figure 1 نمودار همگرایی تعدادی از متغیر های عامل اول

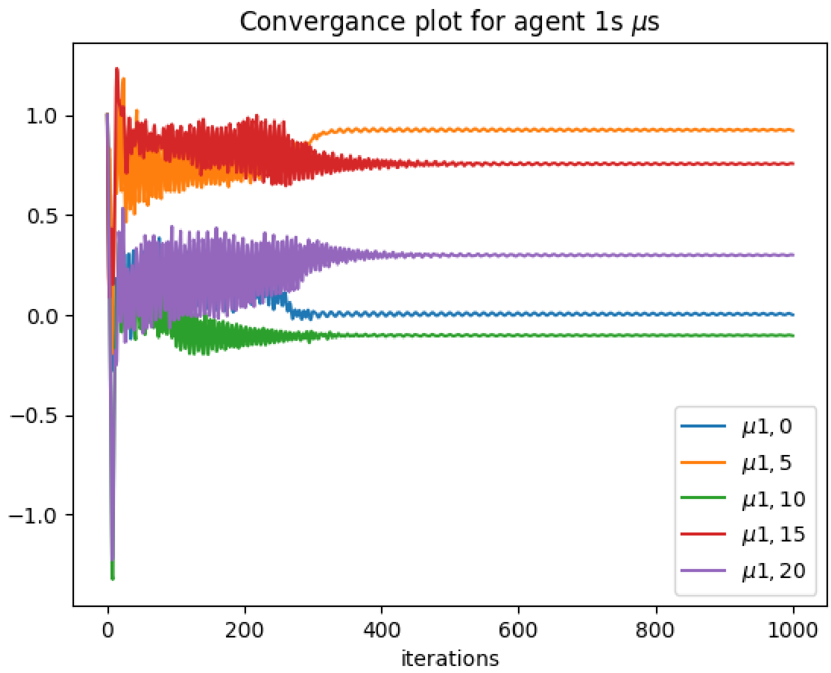

Figure 2 نمودار همگرایی تعدادی از ضرایب لاگراژ داخلی عامل دوم

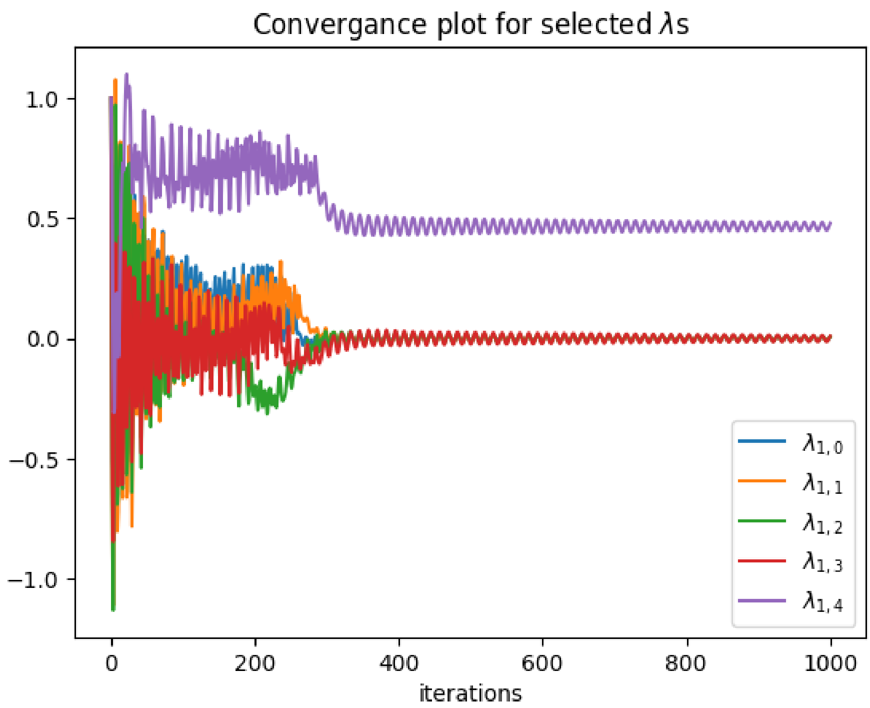

Figure 3 نمودار همگرایی تعدادی از ضرایب لاگراژ خارجی بین عامل اول و دوم

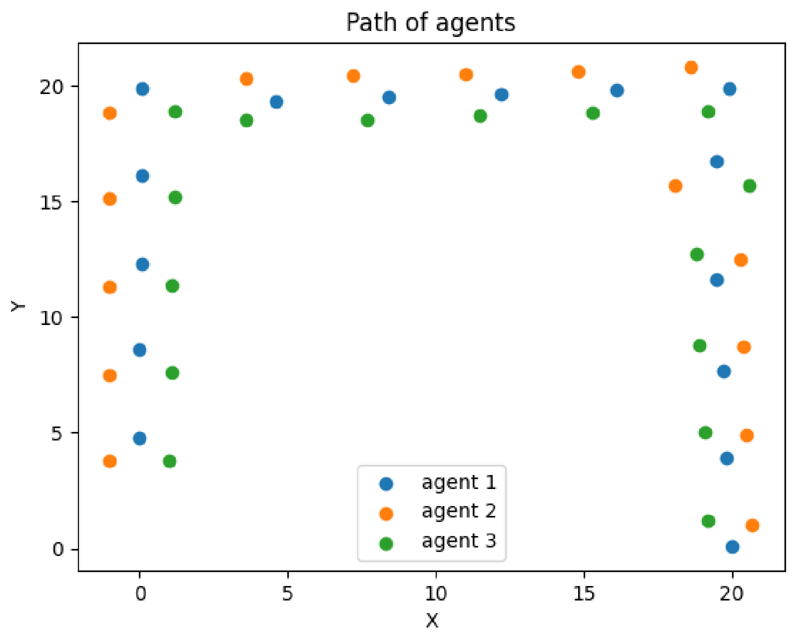

Figure 4 مسیر حرکت عامل ها در فضای 2 بعدی

همانطور که می‌توان دید، روش dual decomposition توانسته مسئله فوق را حل کند. و عامل ها چینش مورد نظر در طول مسیر را پیدا کرده اند. همچنین با نگاهی به نمودار ها می‌تواند دریافت که متغیر های داخلی کاملا همگرا شده ولی ضرایب لاگرانژ با وجود همگرایی قابل قبولی همچنان نوسان میرای کوچکی

دارند که برطرف کردن آن نیاز به پیمایش های طولانی تری دارد. در کل این روش 78 دقیقه به طول انجامید.

نرخ یادگیری lr=0.05/k :

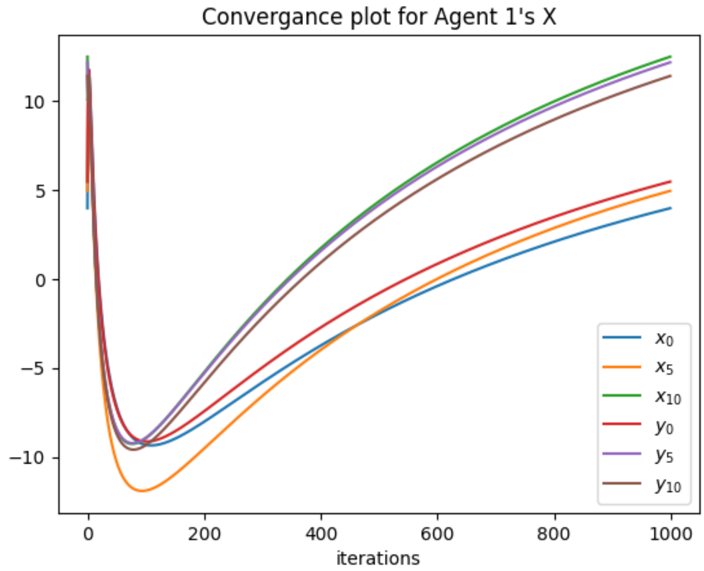

Figure 5 نمودار همگرایی تعدادی از متغیر های عامل اول

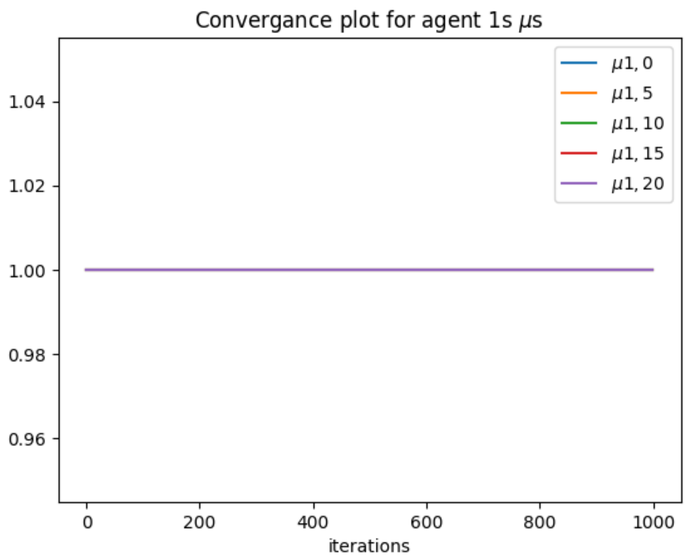

Figure 6 نمودار همگرایی تعدادی از ضرایب لاگراژ داخلی عامل دوم

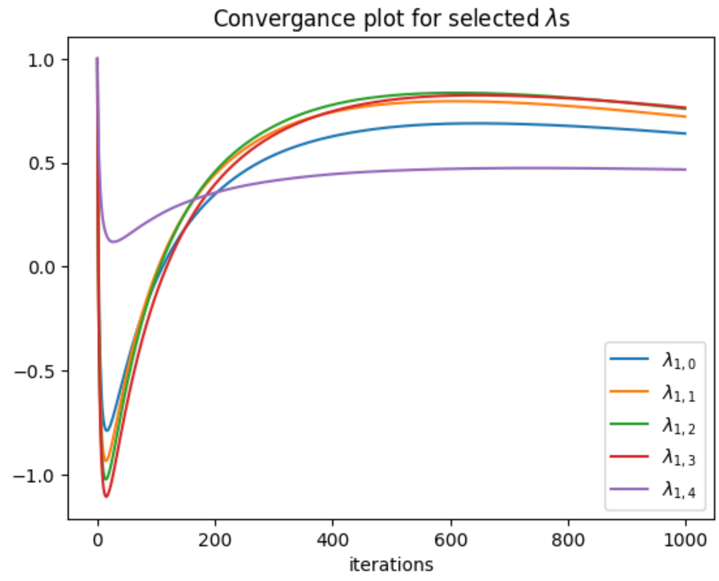

Figure 7 نمودار همگرایی تعدادی از ضرایب لاگراژ خارجی

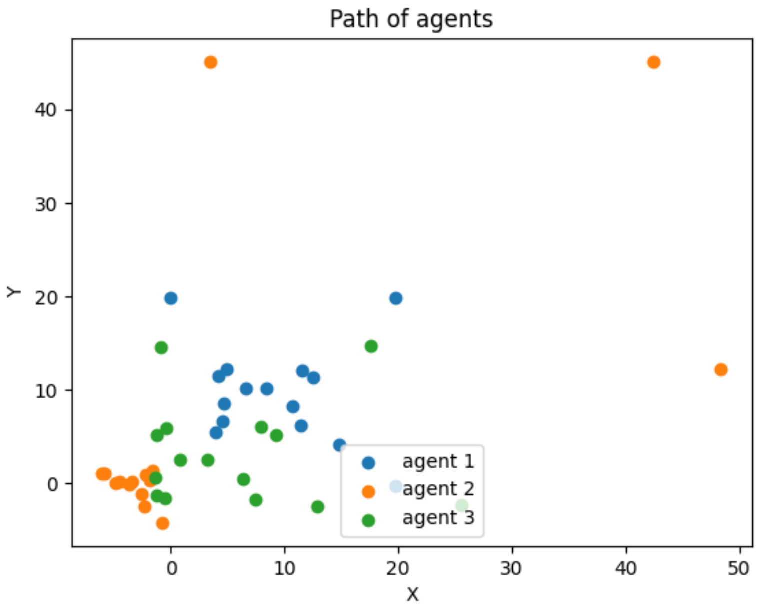

Figure 8 مسیر حرکت عامل ها در فضای 2 بعدی

این روش به دلیل به سرعت کند شدن نرخ یادگیری نتوانسته همگرا شود و بعد از 1000 مرتبه اجرا هنوز در میانه راه است. این روش برای مسئله جواب نداده و مناسب نیست.

نرخ یادگیری $lr = \frac{0.05}{\sqrt{k}}$:

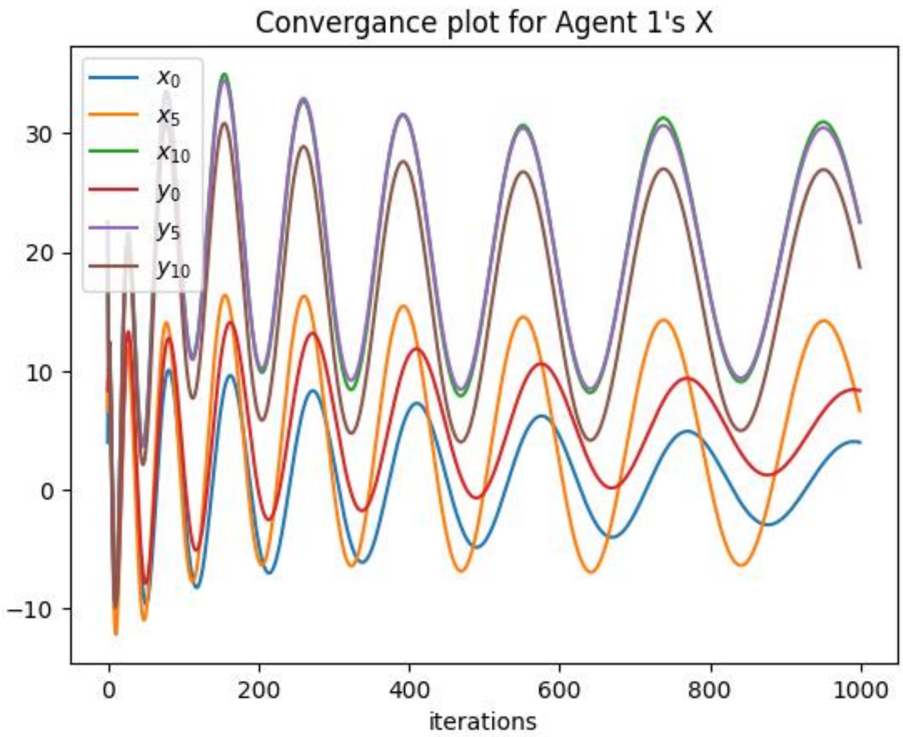

Figure 9 نمودار همگرایی تعدادی از متغیر های عامل اول

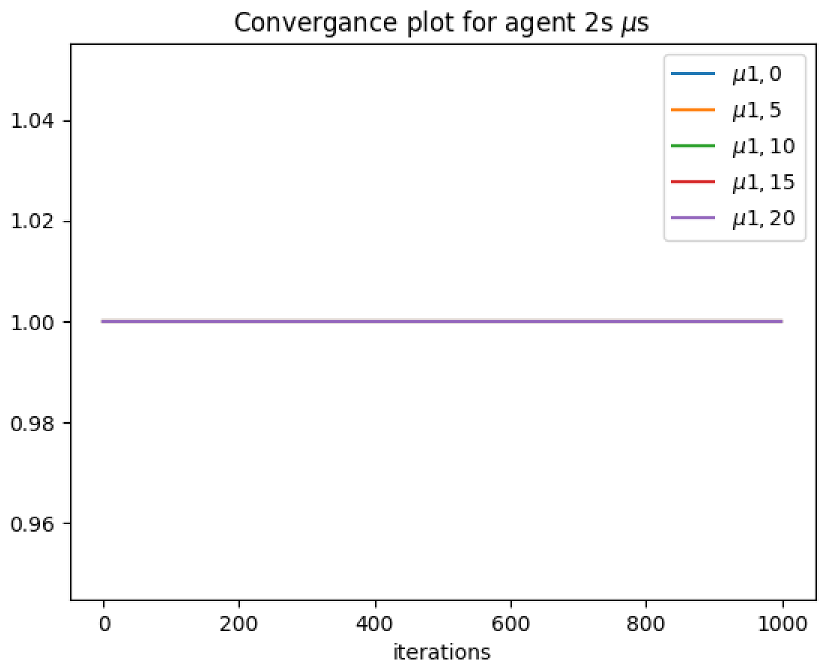

Figure 10 نمودار همگرایی تعدادی از ضرایب لاگراژ داخلی عامل دوم

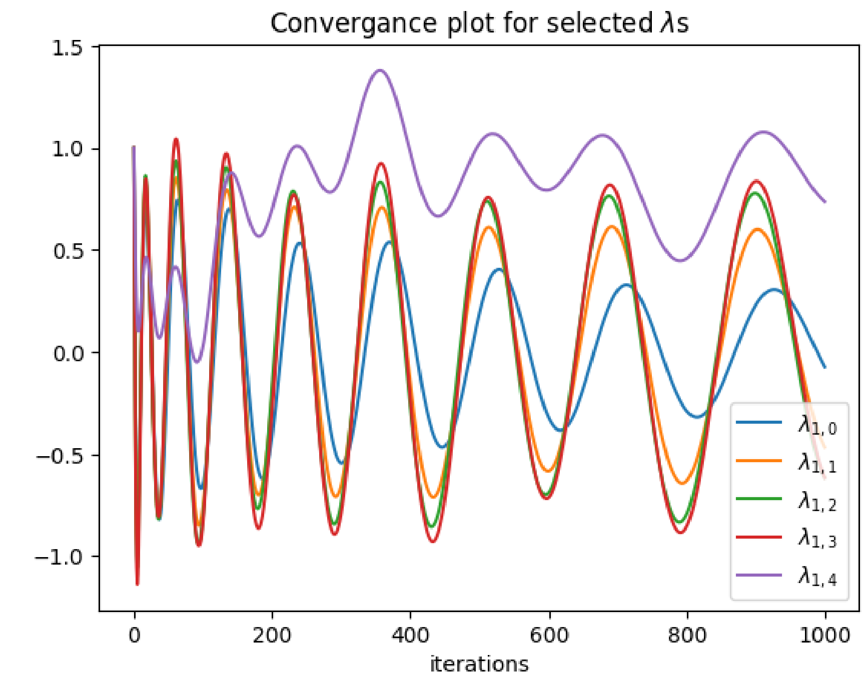

Figure 11 نمودار همگرایی تعدادی از ضرایب لاگراژ خارجی

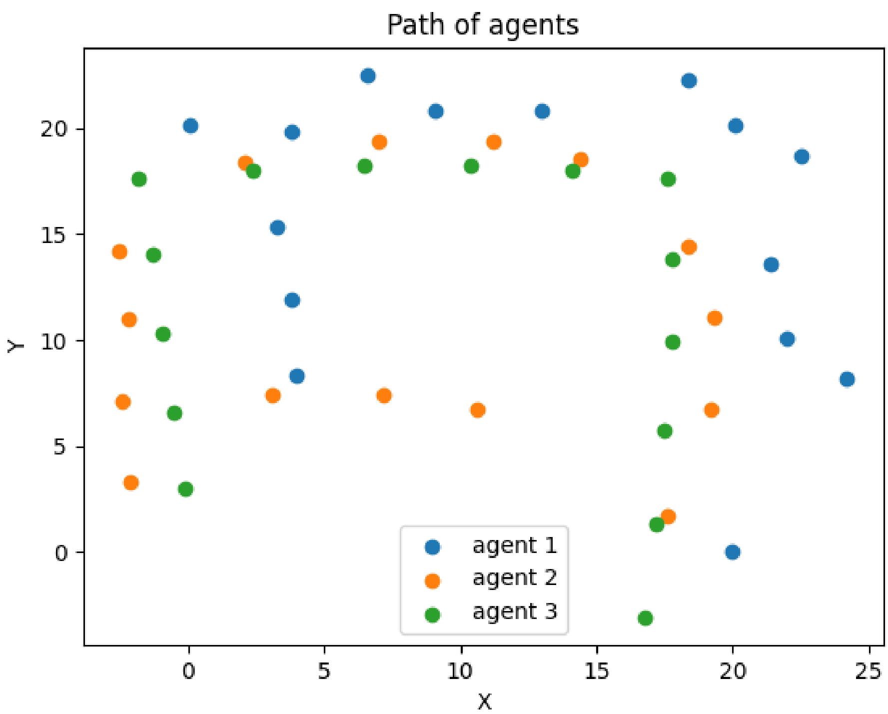

Figure 12 مسیر حرکت عامل ها در فضای 2 بعدی

با نگاهی به نمودار ها می‌توان دید که این نرخ یادگیری با این که بهتر از روش قبلی عمل کرده اما هنوز نتوانسته به همگرایی برسد که علت آن میرایی نرخ یادگیری بوده که بهینه ساز فرصت رسیدن به نقاط بهینه را پیدا نکرده.

طول گام ثابت:

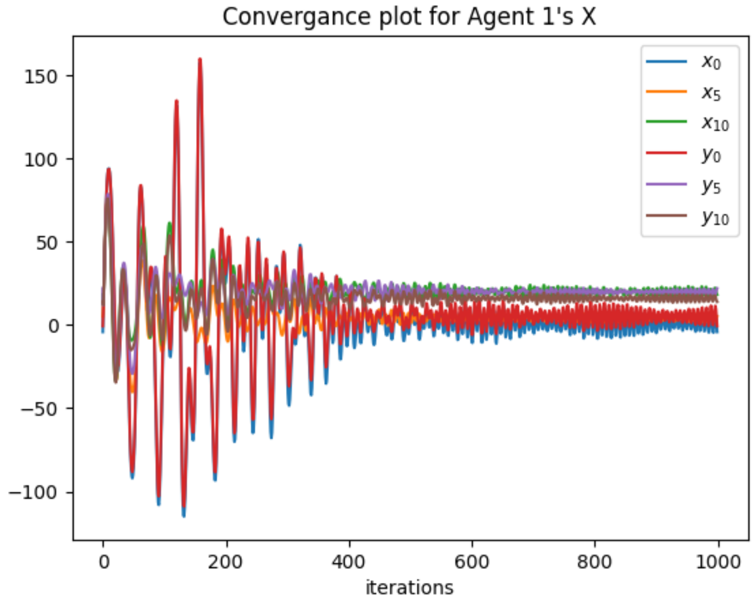

Figure 13 نمودار همگرایی تعدادی از متغیر های عامل اول

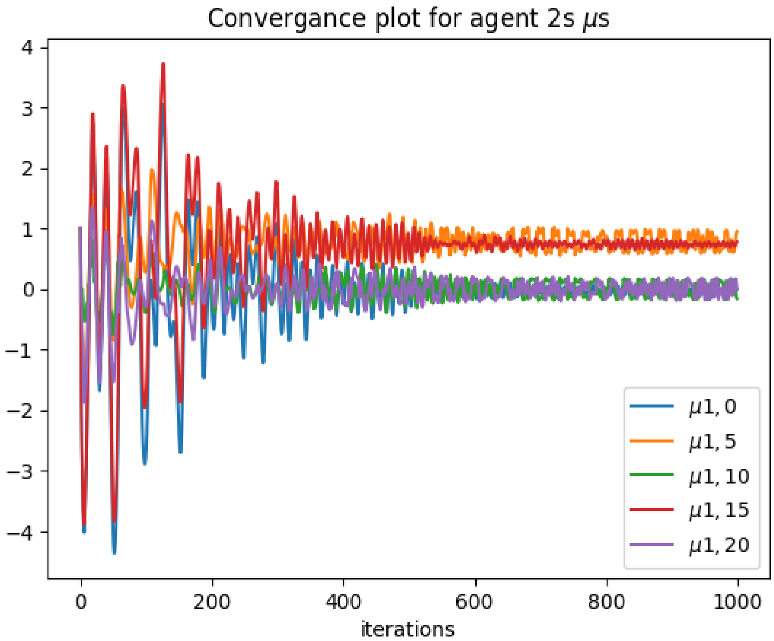

Figure 14 نمودار همگرایی تعدادی از ضرایب لاگراژ داخلی عامل دوم

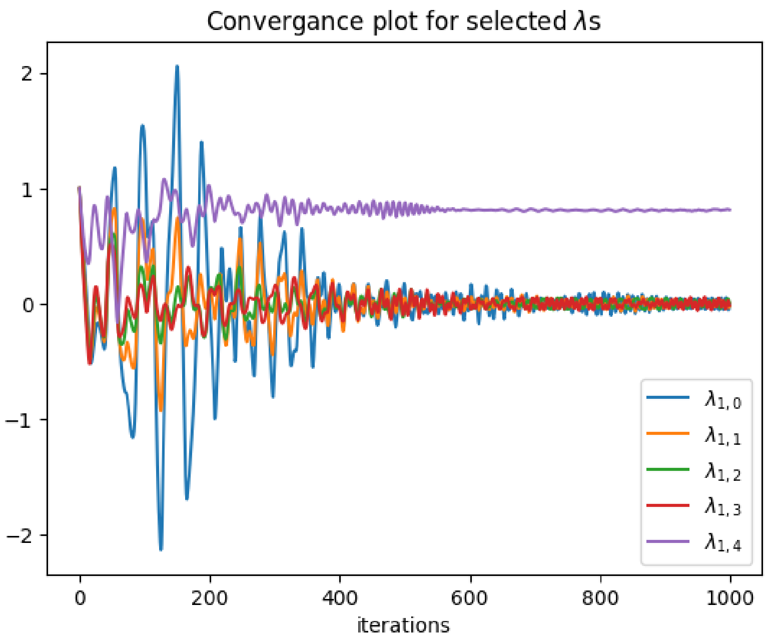

Figure 15 نمودار همگرایی تعدادی از ضرایب لاگراژ خارجی

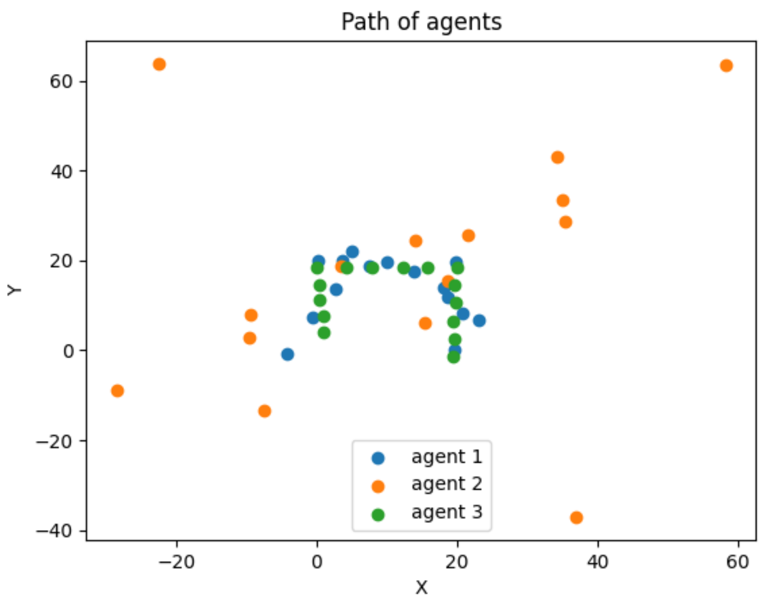

Figure 16 مسیر حرکت عامل ها در فضای 2 بعدی

با نگاهی به نمودار ها می‌توان دید که این روش با این که به ظاهر در نمودار های اول همگرا شده اما به نگاه به مسیر حرکت عامل ها می‌توان دید که همگرایی رخ نداده است. یکی از علت های آن می‌تواند این باشد که در نمودار های قبلی نوسان زیادی در همگرایی وجود داشته و این نوسان باعث شده که نرم گرادیان نیز نوسان داشته باشد و به یک کمینه محلی همگرا شده باشد. این روش حدود 105 دقیقه به طول انجامید

روش ADMM

الف) توضیحات

در این روش بر خلاف روش قبلی آپدیت بصورت ترتیبی در یک سطح انجام می‌شود. همچنین یک عبارت درجه 2 از قید ها به لاگرانژین افزوده می‌شود. که به آن Augmented Lagrangian می‌گویند. روش بدست آوردن و تجزیه کردن دقیقا همانند روش قبل است. (به غیر از عبارت $||K(X)||^2$ که قابل تجزیه نیست و به همان گونه در آپدیت هر کدام از عامل ها بصورت $||K(X)||^2/n_{agent}$ ظاهر می‌شود.

$$AL_i(X_i,\mu_i,\lambda_i) = L_i(X_i,\mu_i,\lambda_i) + \frac{\rho}{2}\left(||D_i(X_i)||^2 + ||K(X)||^2\right)$$

پس روش آپدیت بصورت زیر است:

1. ابتدا مقادیر X,μ,λ را بصورت رندوم مقدار دهی اولیه می‌کنیم.
2. سپس برای 10000 بار آپدیت های زیر را برای هر عامل بصورت ترتیبی انجام می‌دهیم:
   a. For all agents do:
   b. $X_i^{k+1} = X_i^k - \alpha * \frac{\partial}{\partial X_i} AL_i(X,\mu^k,\lambda^l)$
   c. $\mu_i^{k+1} = \mu_i^k + \alpha * \rho * \frac{\partial}{\partial \mu_i} L_i(X^k,\mu,\lambda^l)$
   d. $\lambda_i^{l+1} = \lambda_i^k + \alpha * \rho * \frac{\partial}{\partial \lambda_i} L_i(X^k,\mu^k,\lambda)$

ب) شبیه سازی

مقادیر اولیه شبیه سازی:

$$N = 3, dt = 1\ , \boldsymbol{\rho = 0.1}$$

$$[x_0,y_0] = \left[\begin{bmatrix}0\\1\end{bmatrix}\ \begin{bmatrix}-1\\0\end{bmatrix}\ \begin{bmatrix}1\\0\end{bmatrix}\right]$$

$$checkpoints = \left[\begin{bmatrix}0\\20\end{bmatrix}\ \begin{bmatrix}20\\20\end{bmatrix}\ \begin{bmatrix}20\\0\end{bmatrix}\right]$$

نرخ یادگیری lr = 0.05:

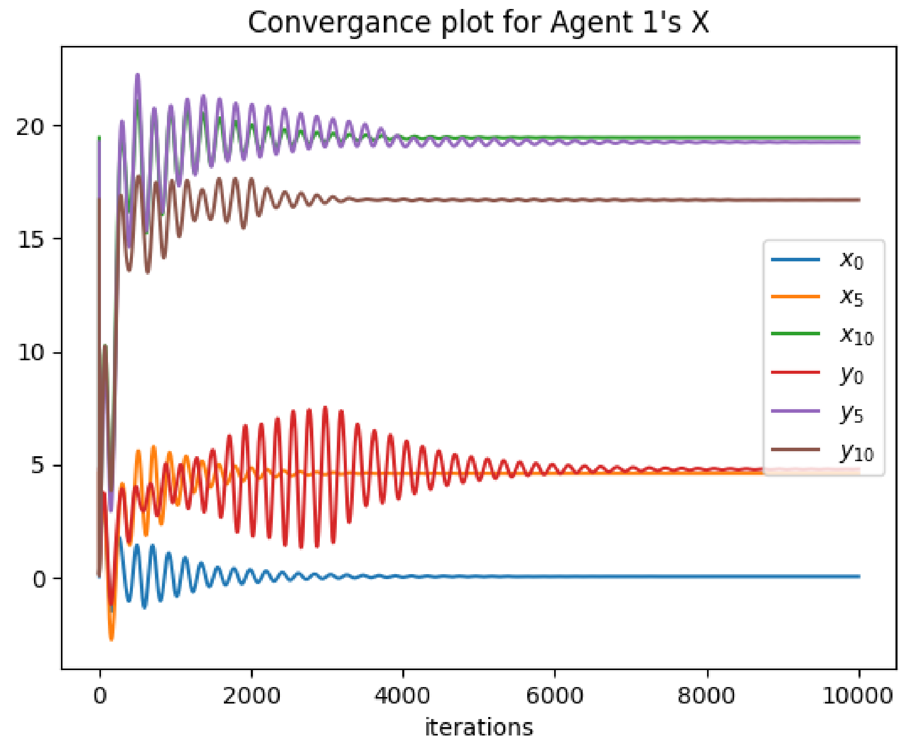

Figure 17 نمودار همگرایی تعدادی از متغیر های عامل اول

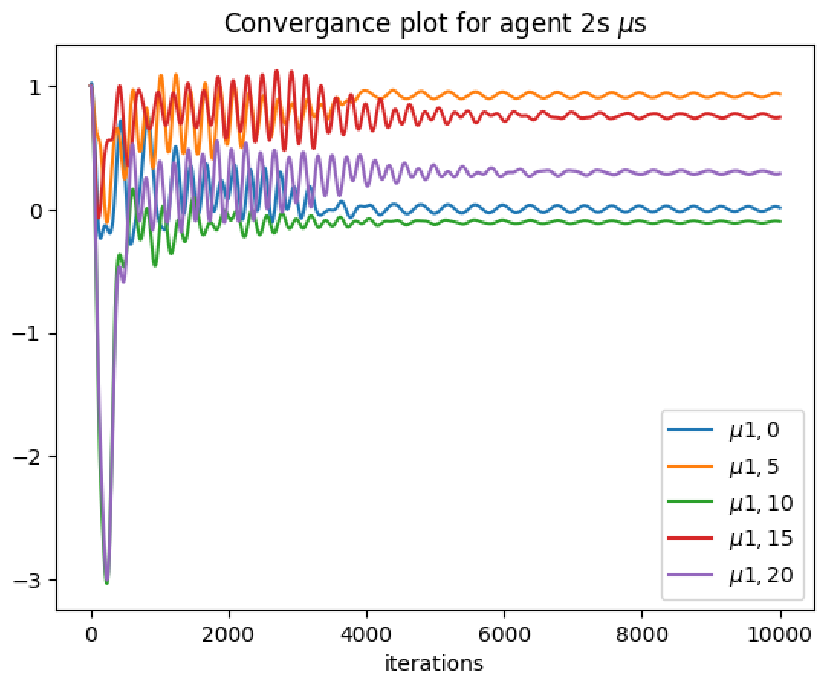

Figure 18 نمودار همگرایی تعدادی از ضرایب لاگراژ داخلی عامل اول

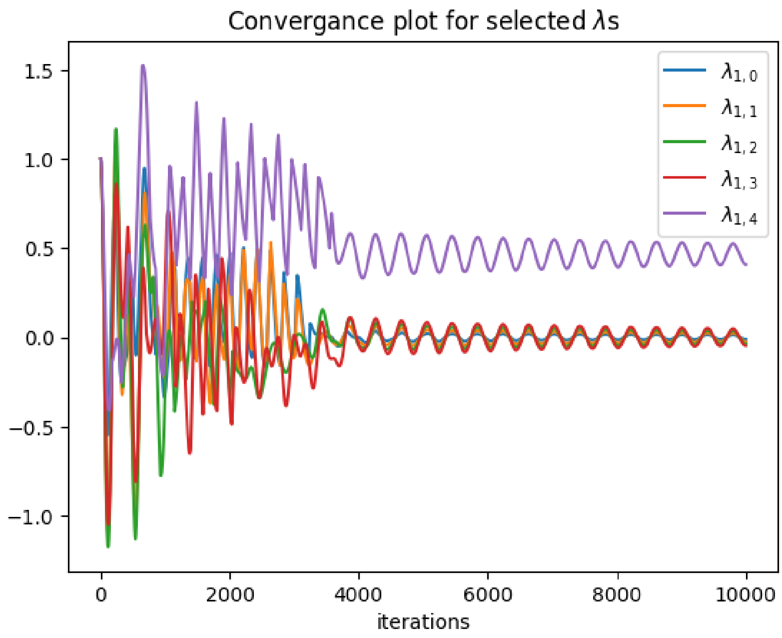

Figure 19 نمودار همگرایی تعدادی از ضرایب لاگراژ خارجی

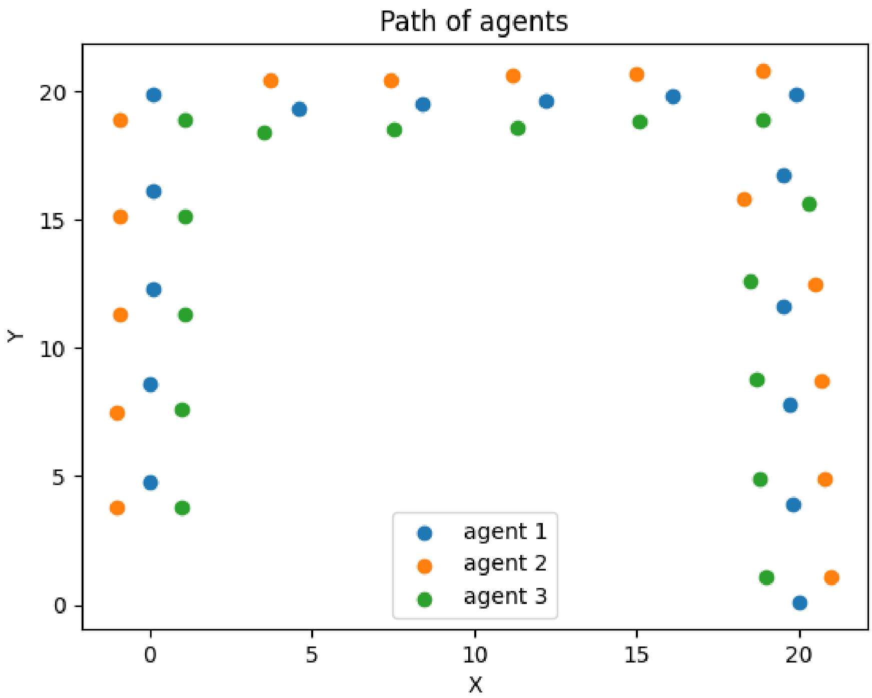

Figure 20 مسیر حرکت عامل ها در فضای 2 بعدی

نرخ یادگیری lr = 0.01:

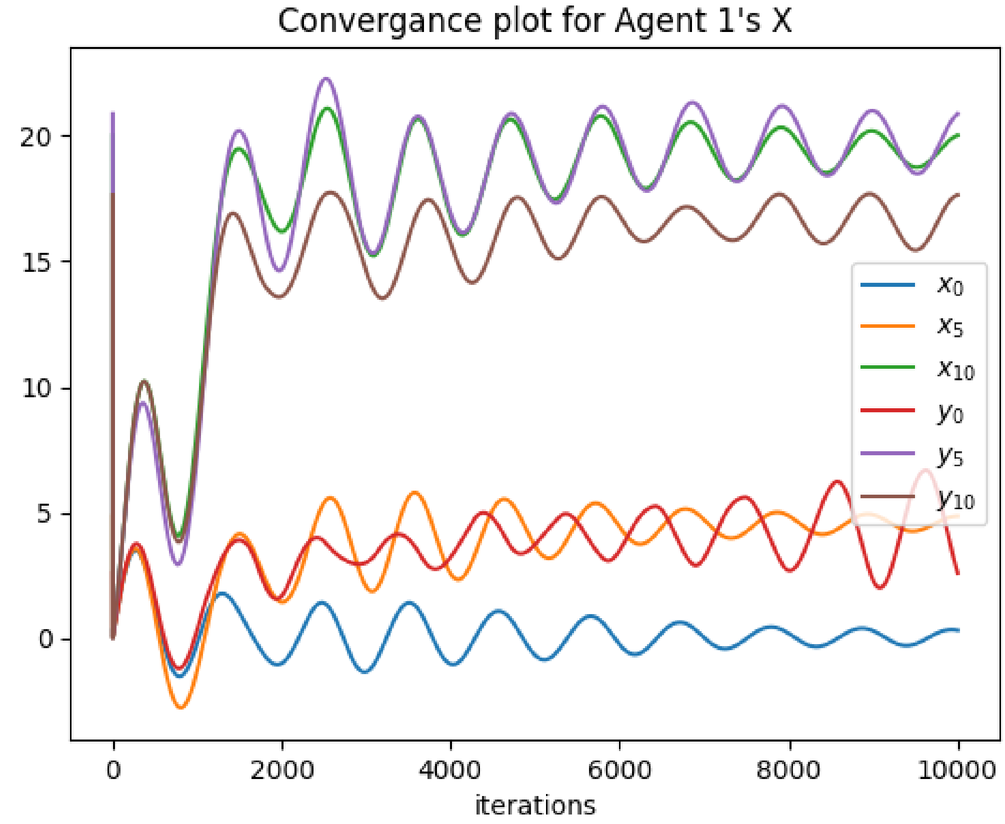

Figure 21 نمودار همگرایی تعدادی از متغیر های عامل اول

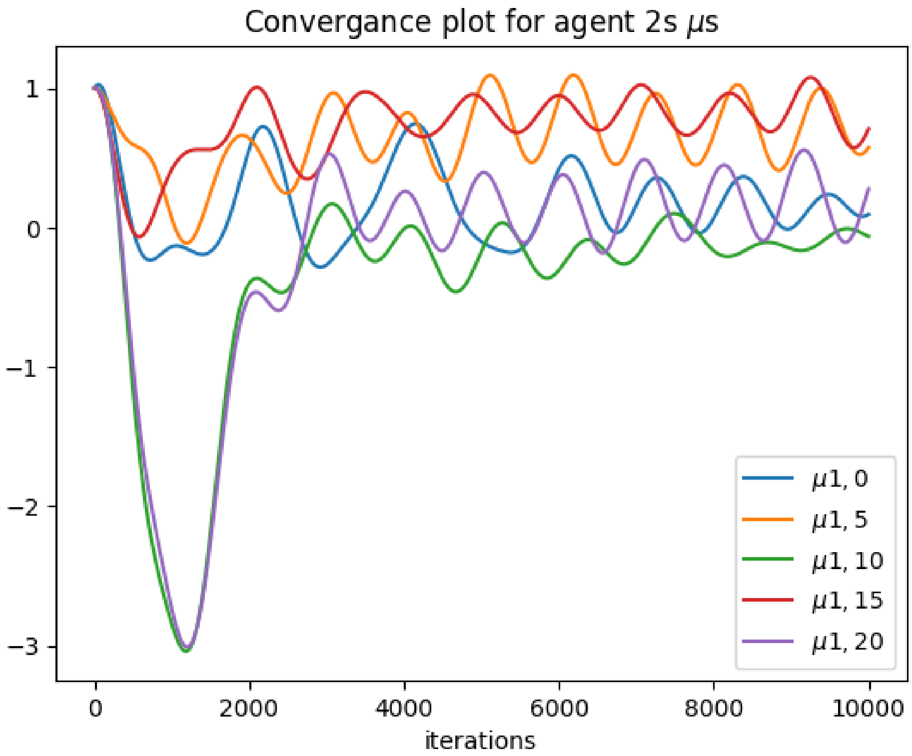

Figure 22 نمودار همگرایی تعدادی از ضرایب لاگراژ داخلی عامل دوم

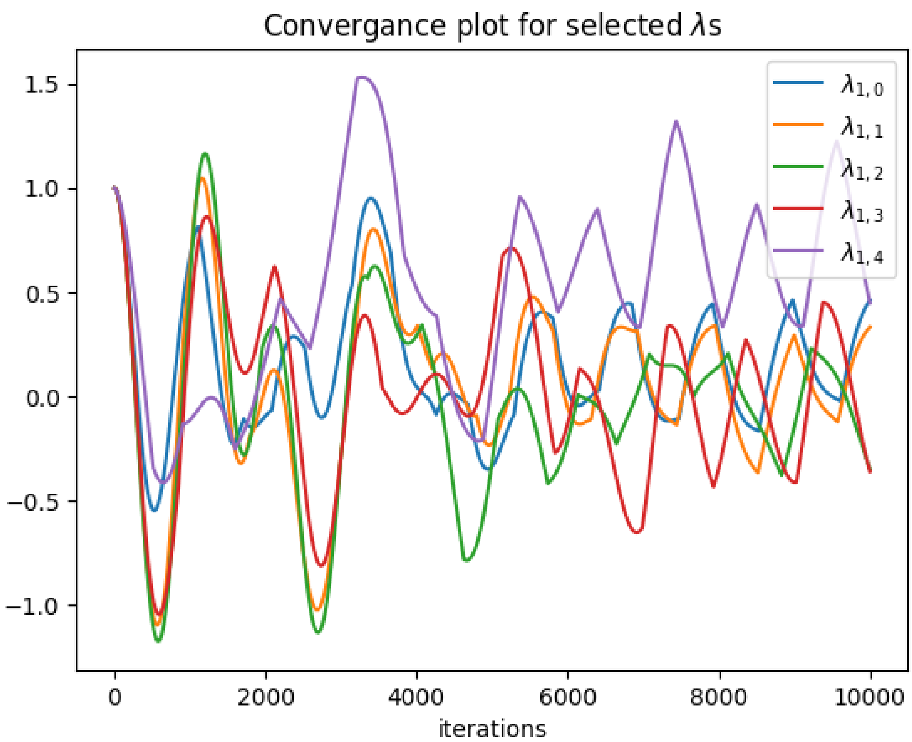

Figure 23 نمودار همگرایی تعدادی از ضرایب لاگراژ خارجی

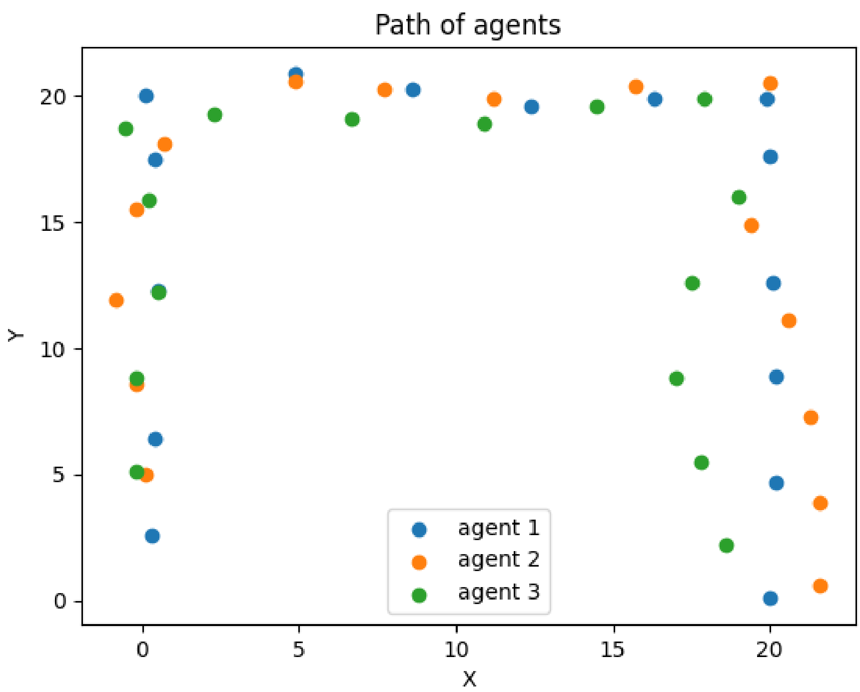

Figure 24 مسیر حرکت عامل ها در فضای 2 بعدی

روش ADMM به خوبی تواسته است که مسئله را حل کند و مقادیر متغیر های همگی میل کرده اند. و مسیر عامل ها مطابق انتظار است. همچنین روند همگرایی آن نسبت به روش dual نوسان کمتری دارد. علت این نوسانات احتمالا روش حل مسئله است که بصورت دو سطحی یا ترتیبی min-max گیری انجام می‌شود. اما با کاهش نرخ یادگیری به 0.01 به همان نسبت همگرایی کمتر رخ داده و هنوز اعوجاج در نمودار ها مشاهده می‌شود که نتیجه آن مسیر معوج عامل ها در شکل آخر است. این روش 75 دقیقه طول کشید.

روش Proximal ADMM

الف) توضیحات

این روش نیز تقریبا مشابه روش ADMM است. با این تفاوت که یک عبارت $\frac{1}{2}\,\|X^k - X^{k-1}\|$ به لاگرانژین اضافه می‌شود. این عبارت باعث می‌شود که همگرایی نرم تری داشته باشیم و نوسان در نمودار متغیر ها کمتر مشاهده شود. پس لاگرانژین جدید بصورت:

$$\boldsymbol{P}AL_i(X_i, \mu_i, \lambda_i) = \boldsymbol{A}L_i(X_i, \mu_i, \lambda_i) + \frac{1}{2}\|X^k - X^{k-1}\|$$

است.

پس روش آپدیت بصورت زیر است:

1. ابتدا مقادیر X, μ, λ را بصورت رندوم مقدار دهی اولیه می‌کنیم.
2. سپس برای 10000 بار آپدیت های زیر را برای هر عامل بصورت ترتیبی انجام می‌دهیم:

a. For all agents do:

b. $X_i^{k+1} = X_i^k - \alpha * \frac{\partial}{\partial X_i} AL_i(X, \mu^k, \lambda^l)$

c. $\mu_i^{k+1} = \mu_i^k + \alpha * \rho * \frac{\partial}{\partial \mu_i} L_i(X^k, \mu, \lambda^l)$

d. $\lambda_i^{l+1} = (1-\tau)\lambda_i^k + \alpha * \rho * \frac{\partial}{\partial \lambda_i} L_i(X^k, \mu^k, \lambda)$

ب) شبیه سازی

مقادیر اولیه شبیه سازی:

$$N = 3, dt = 1\,, \boldsymbol{\rho} = \mathbf{0.1}, \boldsymbol{\tau} = \mathbf{0.05}$$

$$[x_0, y_0] = \left[\begin{bmatrix}0\\1\end{bmatrix} \quad \begin{bmatrix}-1\\0\end{bmatrix} \quad \begin{bmatrix}1\\0\end{bmatrix}\right]$$

$$checkpoints = \left[\begin{bmatrix}0\\20\end{bmatrix} \quad \begin{bmatrix}20\\20\end{bmatrix} \quad \begin{bmatrix}20\\0\end{bmatrix}\right]$$

نرخ یادگیری lr = 0.05:

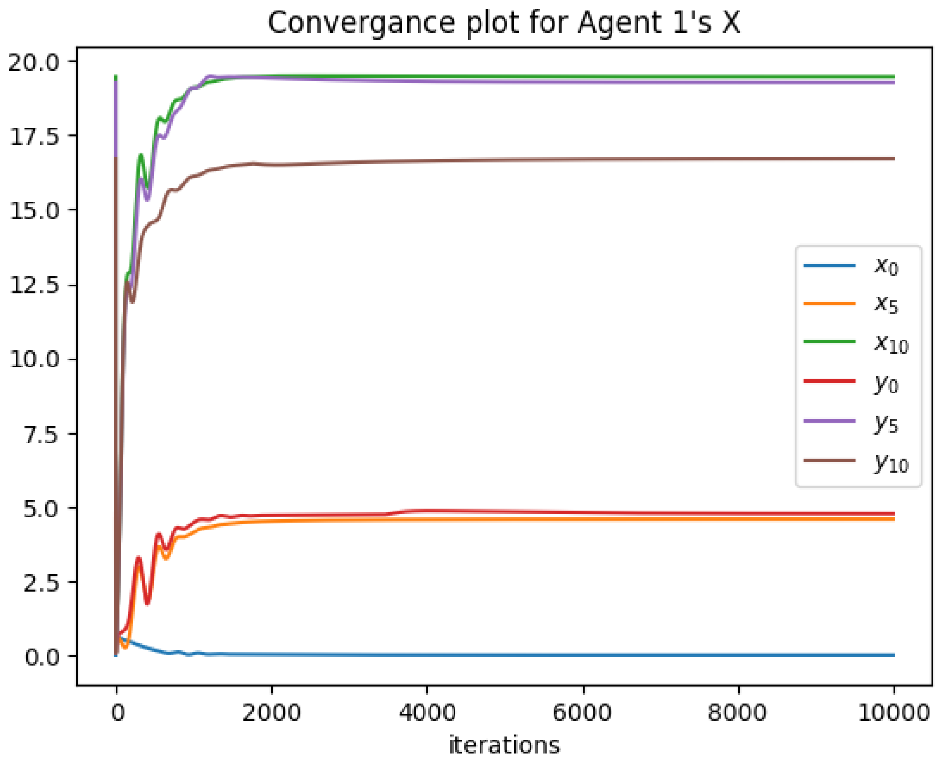

Figure 25 نمودار همگرایی تعدادی از متغیر های عامل اول

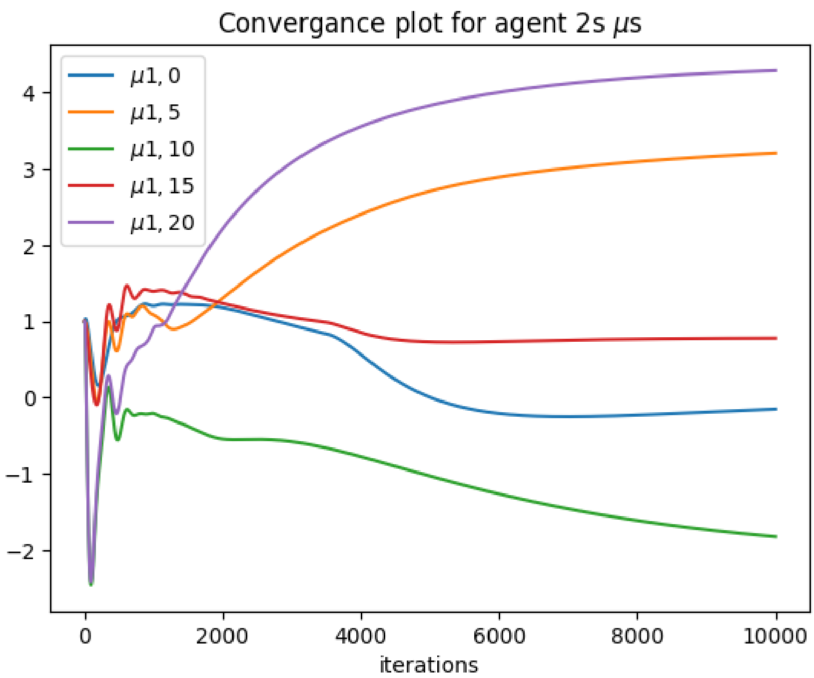

Figure 26 نمودار همگرایی تعدادی از ضرایب لاگراژ داخلی عامل دوم

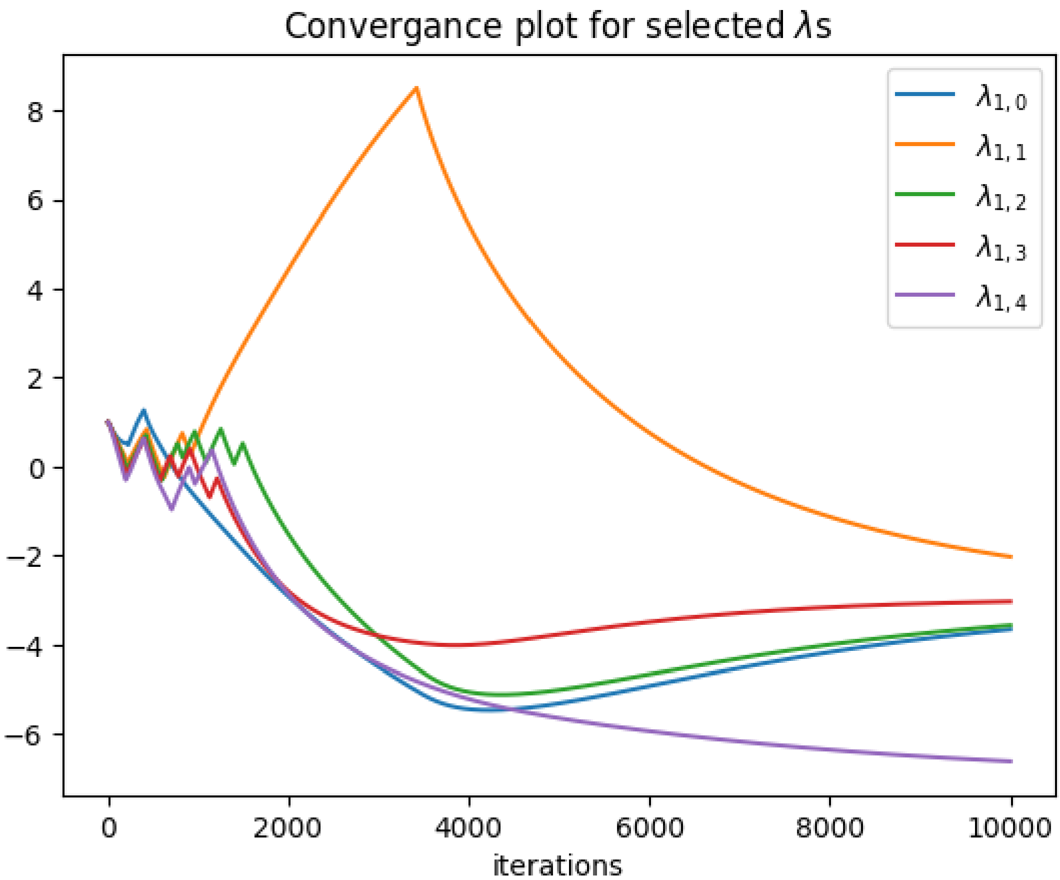

Figure 27 نمودار همگرایی تعدادی از ضرایب لاگراژ خارجی

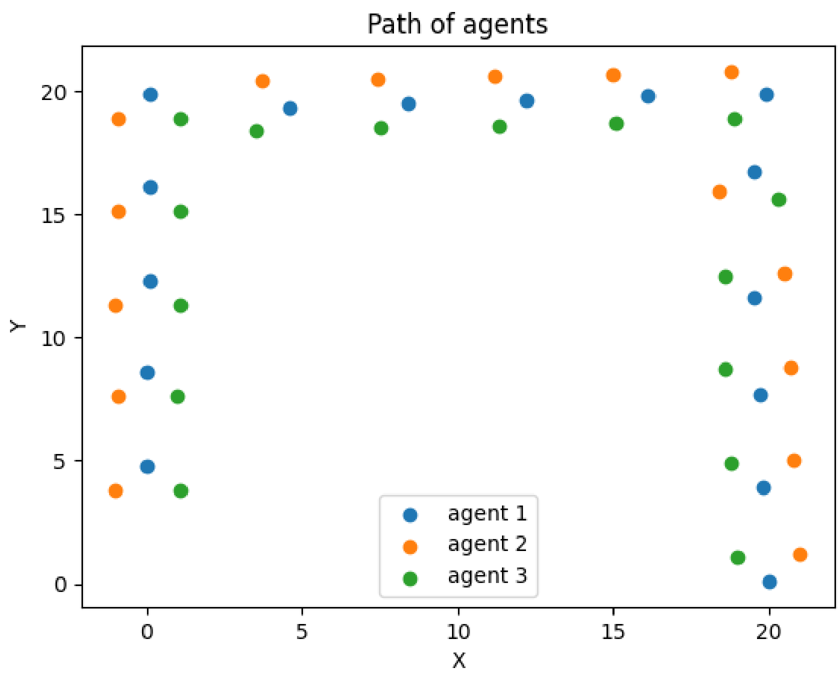

Figure 28 مسیر حرکت عامل ها در فضای 2 بعدی

نرخ یادگیری lr = 0.01:

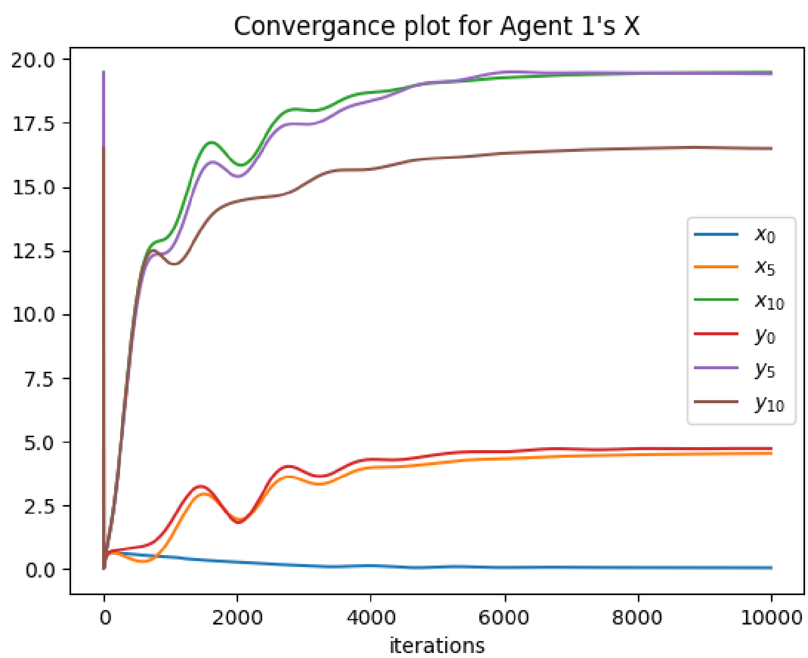

Figure 29 نمودار همگرایی تعدادی از متغیر های عامل اول

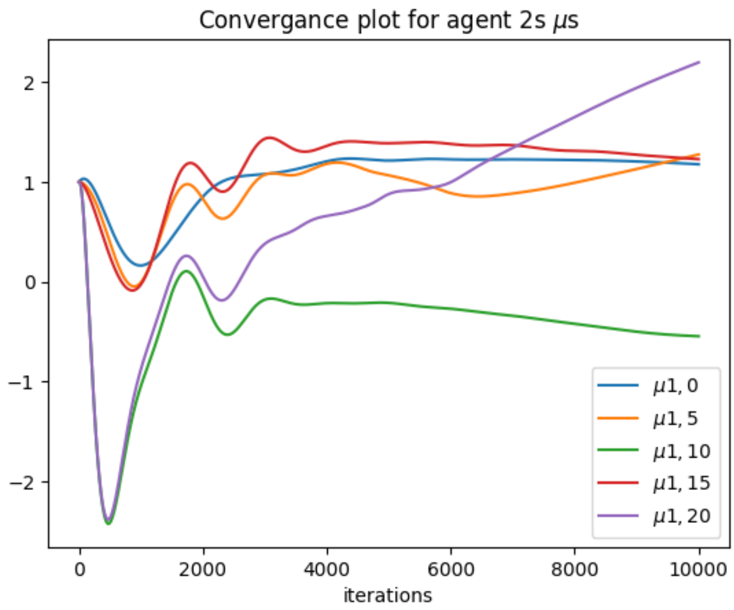

Figure 30 نمودار همگرایی تعدادی از ضرایب لاگراژ داخلی عامل دوم

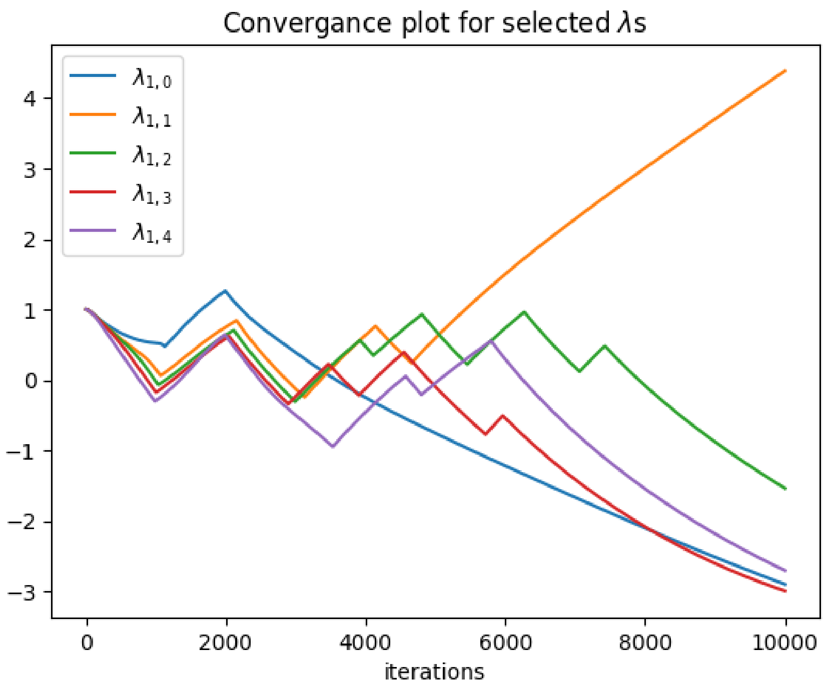

Figure 31 نمودار همگرایی تعدادی از ضرایب لاگراژ خارجی

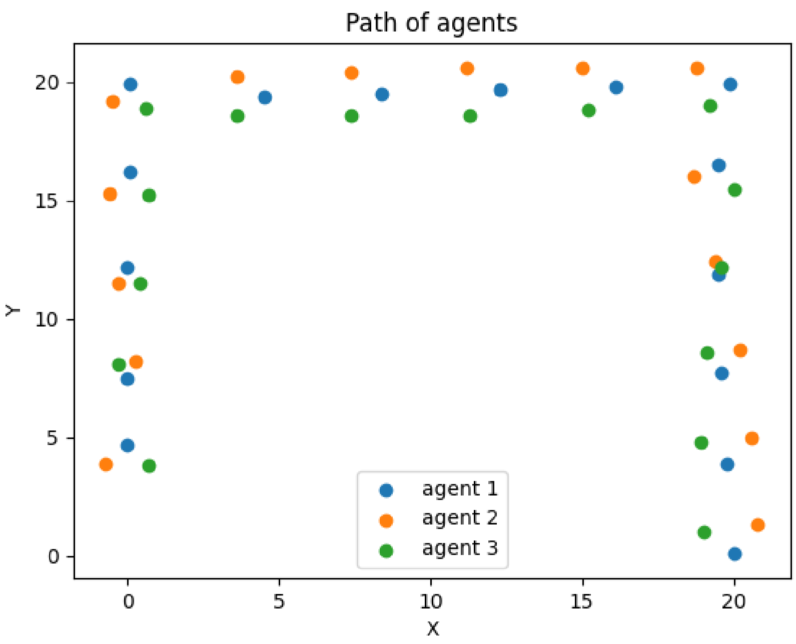

Figure 32 مسیر حرکت عامل ها در فضای 2 بعدی

روش proximal ADMM نیز به خوبی همگرا شده و همگرایی متغیر های آن بسیار زودتر از بقیه روش ها رخ می‌دهد. همچنین نمودار، همگرایی نرم تری دارد که دلیل آن همانطور که در ابتدا گفته شده عبارت مومنتوم اضافه شده به لاگرانژین است. اما حالت دندان اره‌ای در نمودار پدیدار شده که علت آن مشخص نیست. همانند روش قبل با کاهش نرخ یادگیری به همان نسبت همگرایی کمتری داریم. اما روش proximal ADMM نسبت به ADMM در نرخ پایین بهتر همگرا شده است. به دلیل وجو عبارت تکانه. این روش 88 دقیقه طول کشیده است.

در این مسئله روش متمرکز و primal استفاده نشده. دلیل آن کندی بسیار زیاد روش متمرکز است که نیاز به حل یک بهینه سازی 240 متغیر و 120+120 ضریب لاگرانژ دارد.(اما پیاده سازی روش متمرکز بسیار نزدیک به روش dual است و کافی است. که لاگرانژین ها تجزیه نشوند) همچنین به دلیل وجود قیود تساوی تداخلی و نداشتن متغیر تداخلی، استفاده از روش primal تعداد قید ها را 2 برابر و به تعداد قید ها نیز به مسئله متغیر اضافه می‌کرد. و نیاز به تصویر سازی بر روی فضا های سرعت ها و موقعیت های عامل ها بود. به همین دلیل از این 2 روش در مسئله صرف نظر کرده و بجای آن یک مسئله SVM و یک مسئله بدون قید با متغیر تداخلی ساده را با این دو روش حل می‌کنیم.

مسئله SVM

در این مسئله هدف پیدا کردن یک زیر فضا در فضا است که بتواند 2 دسته از داده ها را از هم جدا کند.

$$dataset = \{(x_l, y_l), l \in S_i\}\ datas\ in\ ith\ partition\ of\ dataset.$$

$$\min_{A,b} f(A,b) = \sum_{i=1}^{m} (\frac{1}{m}\|A\|^2 + \sum_{l \in S_i} \max\big(0, 1 - y_l(A^T x_l + b)\big)$$

$$\min_{x} \sum_{i=1}^{m} f_i(x)$$

در روش متمرکز همه دیتاست ها تجمع می‌شوند تا یک دیتاست بدست آید. سپس سابگرادیان گیری بر روی a,b انجام می‌شود تا خط جداساز بهینه بدست آید.

$$X^{k+1} = X^k - \alpha(k) * \frac{\partial}{\partial X} f(X)$$

از 900 داده در دو کلاس با مرکزیت $\left[\begin{bmatrix}2\\2\end{bmatrix}\right], \begin{bmatrix}-2\\-2\end{bmatrix}$ و واریانس 1 استفاده می‌کنیم. A,b را بصورت رندوم مقداردهی می‌کنیم. و سابگرادیان گیری و آپدیت را برای 1000 بار انجام می‌دهیم. (lr=0.01) و نتایج زیر بدست می‌آید:

Figure 33 خط جدا کننده اولیه

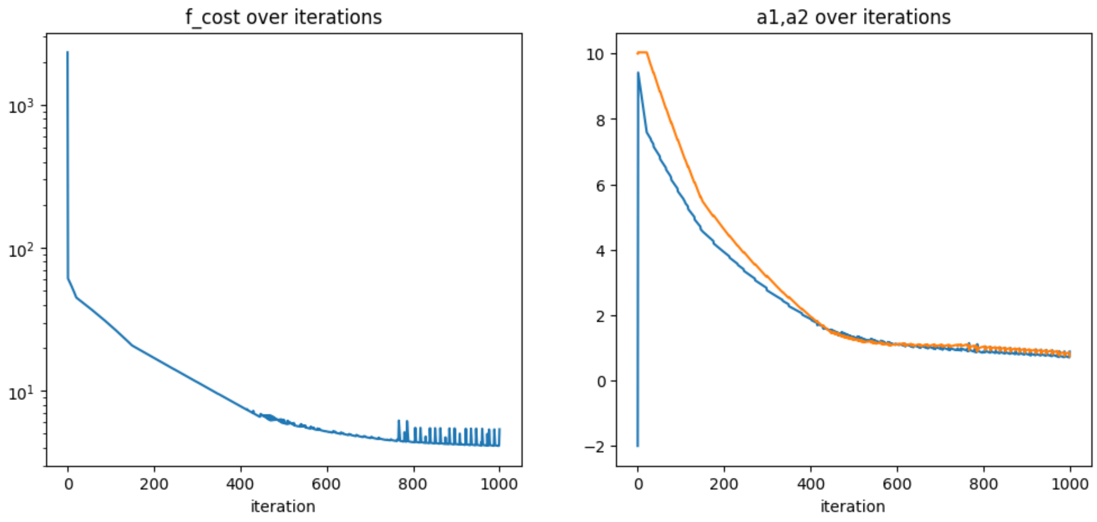

Figure 34 مقدار تابع هزینه و متغیر ها

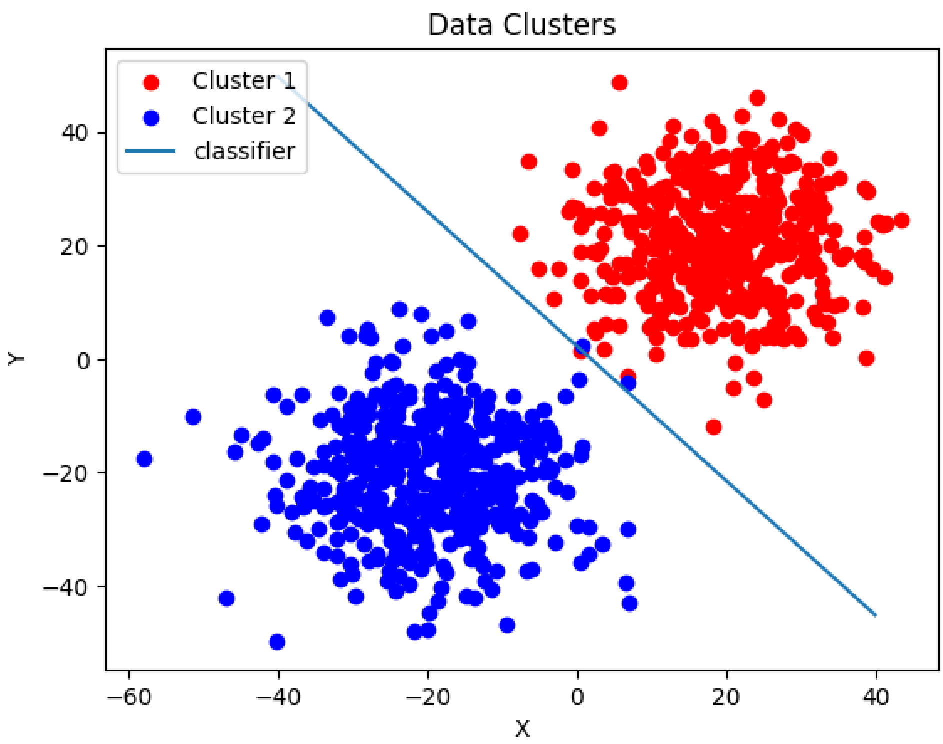

Figure 35 خط بهینه جدا کننده

همانطور که می‌تواند دید خط بهینه توانسته به خوبی داده ها را از هم جدا کند.

روش primal

در این روش دیتاست ها جدا هستند. و به همان تعداد عامل وجود دارد. هر عامل خط بهینه ساز خود را حل می‌کند به همین دلیل خطوط روی هم نمی‌افتند.

$$\min_{A,b} \sum_{i=1}^{m} (\frac{1}{m} \|A\|^2 + \sum_{l \in S_i} \max(0,1 - y_l(A^T x_l + b))$$

$$\min_{x_{1,2,\dots}} \sum_{i=1}^{m} f_i(x_i)$$

$$s.t. \forall i,j \; x_i = x_j$$

$$\forall X_i: X_i^{k+1} = X_i^k - \alpha(k) * \frac{\partial}{\partial X_i} f_i(X_i)$$

$$\forall X_i: X_i^{k+1} = mean(X_i^k)$$

$$n_{agent} = 5, lr = 0.005$$

نتایج بصورت زیر است:

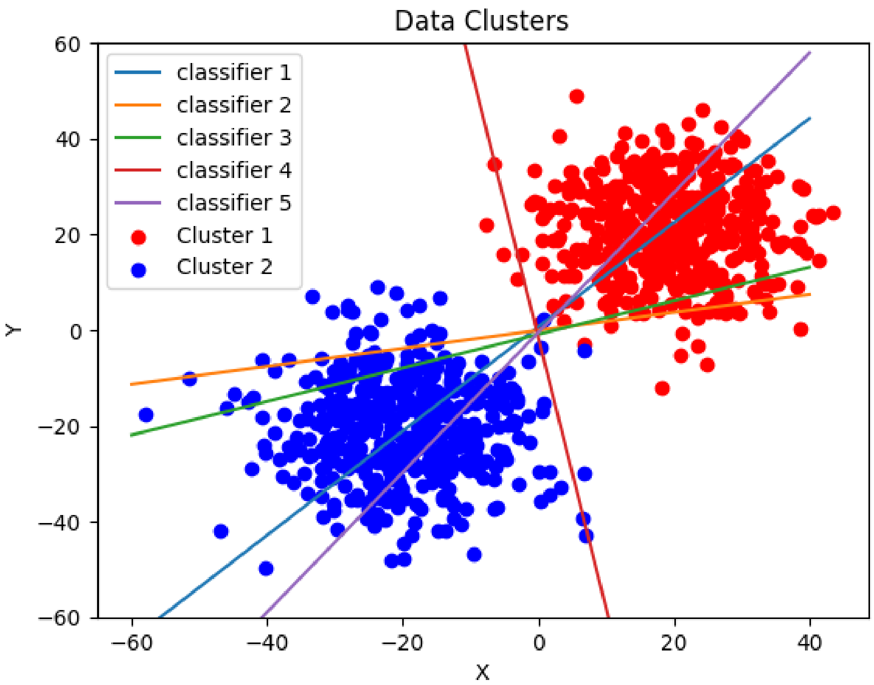

Figure 36 خطوط جدا کننده اولیه

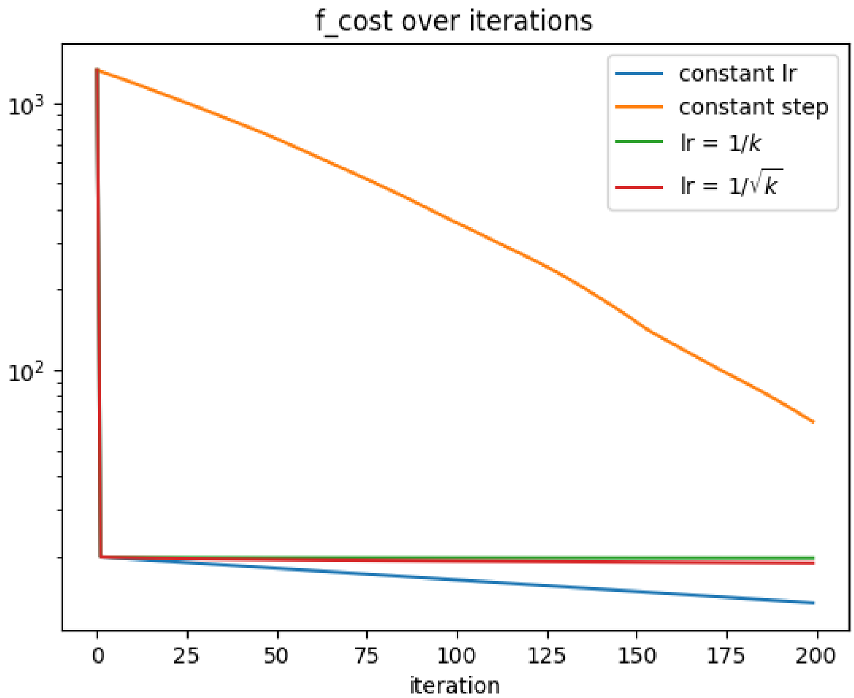

Figure 37 مقدار تابع هزینه با نرخ های یادگیری مختلف

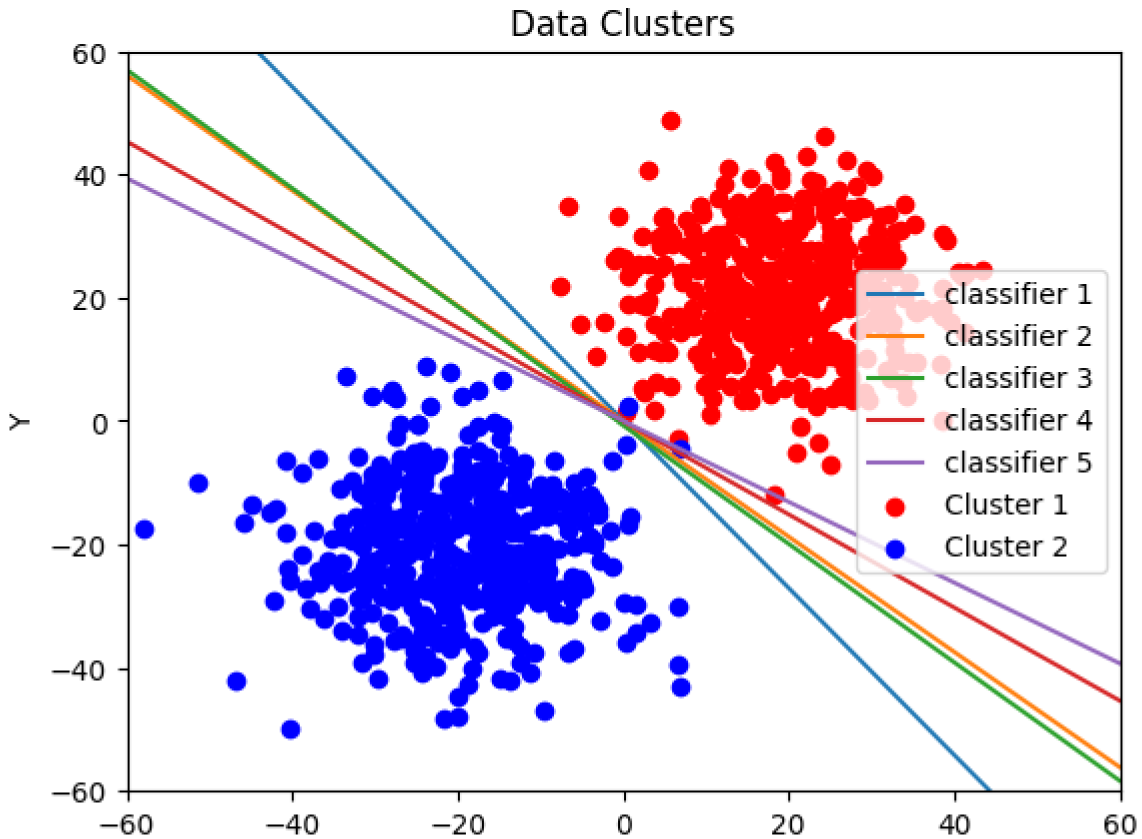

Figure 38 خطوط جدا کننده بهینه هر عامل حالت مربع مجموع نامحدود

با نگاهی به مقدار تابع هزینه می‌توان دید که همه حالات غیر از حالت طول گام ثابت تقریبا عملکرد ثابتی داشته اند. و در سریعا به مقدار بهینه میل کردند اما حالت طول گام ثابت همگرایی کندی نشان داده و نیاز به زمان بیشتری برای همگرایی دارد.

روش ADMM

$$\min_x \sum_{i=1}^{N=5} f_i(x_i)$$
$$s.t. x_i - z = 0$$

$$L_i(x,z,y) = f_i(x_i) + y_i^T(x_i - z) + \frac{\rho}{2}\|x_i - z\|$$

ADMM

1. Initialize x,y,z
2. $x_i^{k+1} = argmin\ L_i(x_i, y^k, z^k)$
3. $z^{k+1} = \frac{1}{N}\sum(x_i(k+1) + \frac{y_i^k}{\rho}$
4. $y_i^{k+1} = y_i^k + \rho(x_i^{k+1} - z^{k+1}$

نتایج

$$k = 200, \rho = 50$$

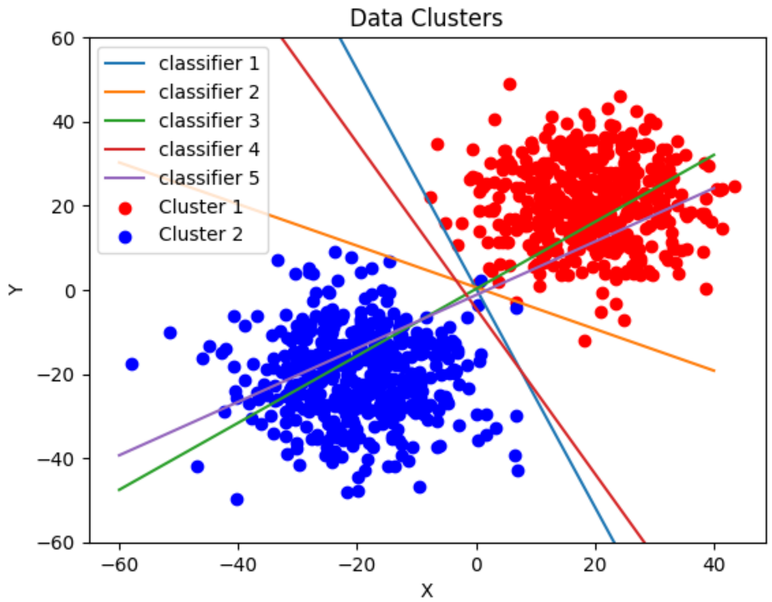

Figure 39 دسته ها و خطوط جدا کننده اولیه

Figure 40 نمودار تابع هزینه

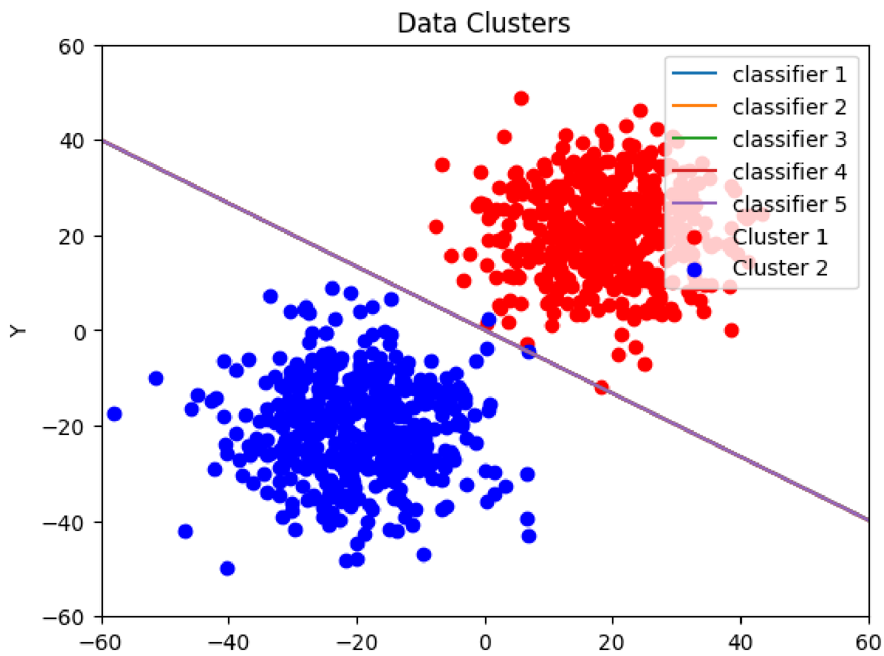

Figure 41 دسته و خطوط جدا کننده بعد از میل کردن به روش **lr** ثابت

با نگاهی به نمودار های بالا می‌توان دید که روش ADMM مسئله بهینه سازی و اجماع را به خوبی حل کرده و خطوط جدا کننده همگرا شدند و به خوبی 2 کلاس را جدا کرده اند.

مسئله با متغیر تداخلی

برای مشاهده کاربرد های دیگری از مسئله primal یک مسئله با قید تداخلی حل می‌کنیم. مسئله شامل 3 تابع هدف هستند که 2 متغیر اول آن ها داخلی و 2 متغیر دوم آن ها مشترک هستند.

$$f(x_1 x_2, x_3, y) = f_1(x_1, y) + f_2(x_2, y) + f_3(x_3, y)$$

$$f_1(x_1, y) = 2(x_1[1]-1)^2 + (x_1[2]-y[0])^2 + (y[0]-1)^2 + 2.5(y[1]+2)^2$$

$$f_2(x_2, y) = (x_2[1]-1)^2 + 2*(x_1[2]-3)^2 + 2*(y[0]-x_1[2])^2 + 0.5(y[2]-2)^2$$

$$f_3(x_3, y) = (x_3[1]-1)^2 + (x_3[2]-3)^2 + 2*(y[0]-x_3[2])^2 + 0.5(y[2]+x_3[1])^2$$

این مسئله را ابتدا بصورت متمرکز حل می‌کنیم. روش حل مشابه قسمت های قبل است.

Lr = 0.05

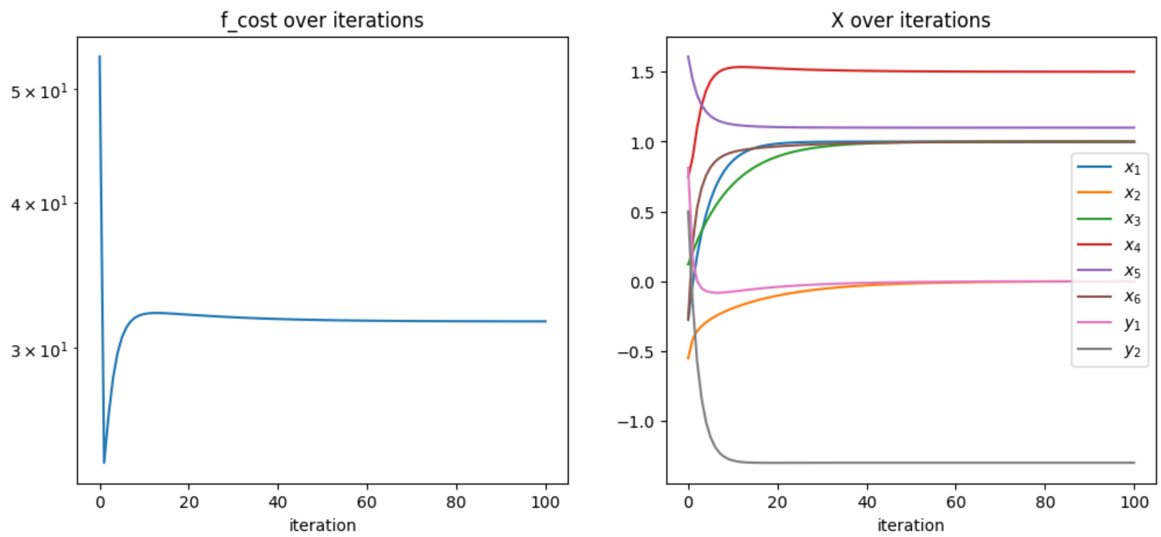

Figure 42 مقدار تابع و متغیر ها

روش متمرکز مسئله را به راحتی حل کرد

حال برای روش primal همانند آنچه برای روش 2 سطحی dual مسئله اصلی توضیح داده شد پیش می‌رویم. یعنی هر عامل تابع هزینه خودش را نسبت به متغیر های داخلی خودش آپدیت می‌کند. سپس در سطح بالاتر متغیر تداخلی y آپدیت می‌شود.

1. Initialize X, Y
2. For 100 times do:
   a. For all agents do:
      i. For 100 times do:
         1. $x_i^{k+1} = x_i^k - \alpha(l) * \frac{\partial}{\partial x_i}(f_i(x_i, y^l)$

b. $y_i^{l+1} = y_i^l - \alpha(l) * \frac{\partial}{\partial y_i}(f_i(x_i^l, y))$

با پیاده سازی روش فوق نتایج زیر بدست می‌آید.

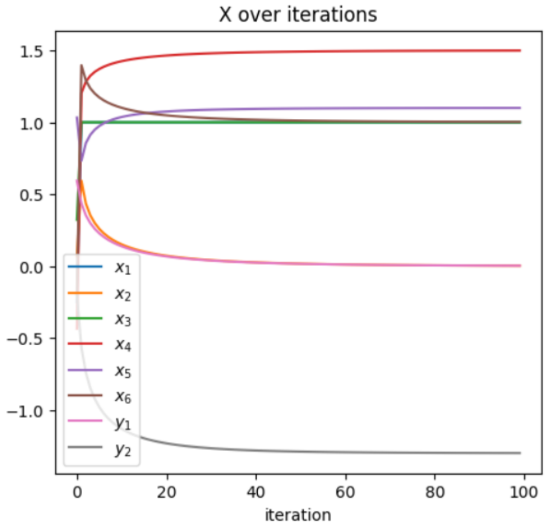

Figure 43 مقادیر متغیر ها

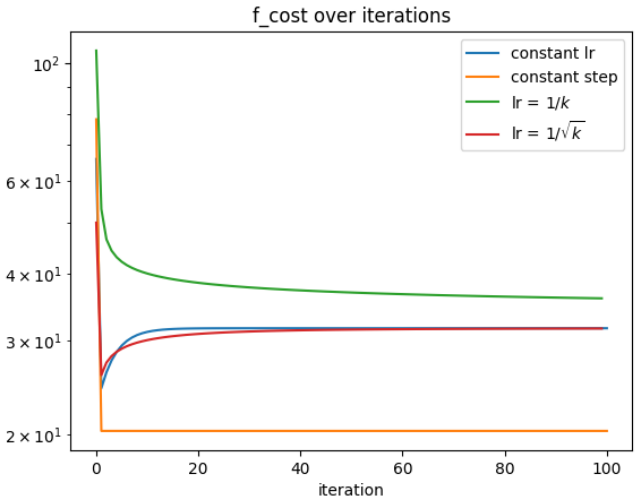

Figure 44 مقدار تابع هزینه با **lr** های مختلف

با نگاهی به شکل می‌تواند دید که بهترین عملکرد را طول گام ثابت دارد. که به سرعت همگرا شده. بعد از آن روش lr ثابت و روش $\frac{1}{\sqrt{k}}$ مشابه هم عملکرده و روش $\frac{1}{k}$ نیز از همه کندتر بوده.

اما اگر در نظر بگیریم که روش متمرکز نیز از lr ثابت استفاده کرده است. آنگاه جواب هر 2 روش ثابت است. اما روش primal به دلیل 2 سطحی بودن نیاز به محاسبه بیشتر دارد.