TALBI ABDELLAH iid2

Q1

initialisation : initialiser une population des solution potentielles créée aléatoirement. Chaque solution est représentée comme un ensemble des gènes.

Évaluation : Chaque solution de la population est évaluée en fonction de son aptitude (ou de sa valeur) par rapport au problème donné.

Sélection : Les solutions les plus adaptées (celles qui ont les meilleures valeurs d'aptitude) sont sélectionnées pour la reproduction. Croisement (reproduction) : Les solutions sélectionnées se reproduisent en échangeant des parties de leurs gènes pour créer de nouvelles solutions (appelées "descendants").

Mutation : Les gènes des descendants nouvellement créés peuvent subir des mutations aléatoires avec une faible probabilité. Cela ajoute une diversité à la population et aide à éviter la convergence prématurée vers une solution suboptimale.

Remplacement : Les descendants et/ou une partie de la population précédente sont sélectionnés pour former la nouvelle population. Cela garantit que la population suivante contient à la fois des solutions améliorées et des solutions potentiellement nouvelles.

Critères d'arrêt : L'algorithme génétique s'exécute généralement pendant un certain nombre d'itérations ou jusqu'à ce qu'un critère d'arrêt prédéfini soit atteint. Les critères d'arrêt courants peuvent être le nombre maximal d'itérations, l'obtention d'une solution satisfaisante ou l'absence d'amélioration significative de la population.

Q2

--Différences entre les algorithmes génétiques et les algorithmes de descente par gradient :

Représentation des solutions : Les algorithmes génétiques utilisent généralement une représentation sous forme de population de solutions potentielles, où chaque solution est généralement une combinaison de gènes ou de caractéristiques. En revanche, les Algorithmes de de descente de Gradient travaillent généralement avec une seule solution à la fois, souvent représentée par un vecteur de paramètres.

Exploration de l'espace de recherche : Les algorithmes génétiques sont des méthodes stochastiques qui explorent l'espace de recherche de manière globale. Ils utilisent des opérateurs tels que la sélection, le croisement et la mutation pour générer une diversité de solutions. En revanche, les algorithmes de descente de gradient sont des méthodes déterministes qui se déplacent généralement dans l'espace de recherche en suivant le gradient de la fonction objectif. Ils visent à trouver localement le minimum ou le maximum de cette fonction.

Convergence : Les algorithmes génétiques peuvent trouver des solutions globales potentiellement meilleures, mais ils nécessitent souvent plus d'itérations pour converger vers une solution optimale. Les algorithmes de descente de gradient peuvent converger plus rapidement vers un minimum local, mais ils peuvent être piégés dans des optima locaux et ne pas trouver la meilleure solution globale.

Adaptabilité à la nature du problème : Les algorithmes génétiques sont souvent utilisés pour résoudre des problèmes d'optimisation combinatoire ou des problèmes où la fonction objectif est difficile à définir ou à dériver. Ils sont également utiles pour explorer un grand espace de recherche. Les algorithmes de descente de gradient sont plus adaptés aux problèmes d'optimisation différentiables, où la fonction objectif et les gradients peuvent être calculés.

--Points communs entre les algorithmes génétiques et les algorithmes de descente par gradient :

Recherche de solutions optimales : Les deux types d'algorithmes visent à trouver des solutions optimales à un problème donné, bien que leurs approches diffèrent.

Itérations et amélioration itérative : Les algorithmes génétiques et les algorithmes de descente de gradient utilisent tous deux des itérations pour améliorer progressivement les solutions. Dans le cas des algorithmes génétiques, chaque itération est généralement appelée une génération.

Utilisation d'une fonction objectif : Les deux types d'algorithmes nécessitent une fonction objectif qui évalue la qualité des solutions proposées.

Paramètres d'optimisation : Les algorithmes génétiques et les algorithmes de descente de gradient comportent des paramètres d'optimisation tels que la taille de la population, les taux de mutation, les critères d'arrêt, etc., qui doivent être réglés pour obtenir de bons résultats.

Q3

L'analyse de convergence d'un algorithme génétique (AG) est cruciale pour déterminer son efficacité et ses performances. Voici quelques techniques courantes pour examiner la convergence d'un AG :

Suite à l'évolution des valeurs d'aptitude : Le suivi de l'évolution des valeurs d'aptitude de la meilleure solution trouvée au fil des générations est l'une des principales mesures de convergence. Pour visualiser la convergence, cette évolution peut être représentée graphiquement.

Convergence vers une solution optimale connue : Si vous avez une solution optimale connue pour votre problème, vous pouvez comparer les solutions créées par l'AG à cette solution de référence. Une convergence se produit lorsque les solutions se rapprochent progressivement de la solution optimale connue.

Convergence vers une solution optimale connue : Si vous avez une solution optimale connue pour votre problème, vous pouvez comparer les solutions créées par l'AG à cette solution de référence. Une convergence se produit lorsque les solutions se rapprochent progressivement de la solution optimale connue.

Étude des courbes de convergence : Vous pouvez observer les tendances globales et les schémas de convergence de l'AG en examinant les courbes de convergence. Par exemple, vous pouvez examiner la vitesse de convergence, l'apparition de plateaux, les fluctuations ou les sauts brusques dans les valeurs d'aptitude. Ces informations vous aident à comprendre comment l'algorithme progresse et s'il converge efficacement.

Analyse statistique : L'analyse statistique peut fournir des informations plus approfondies sur la convergence AG. Vous pouvez utiliser des intervalles de confiance pour évaluer la stabilité des résultats, effectuer des tests statistiques pour comparer les performances de différentes exécutions de l'algorithme ou utiliser des techniques d'analyse de variance des paramètres de l'AG sur la convergence.

Q4

Les algorithmes d'optimisation utilisent souvent l'intensification et la diversification pour trouver la meilleure solution possible à un problème donné. --l'intensification signifie concentrer les efforts de recherche sur les solutions les plus prometteuses ou les zones les plus prometteuses de l'espace de recherche. Cela signifie que l'algorithme se concentre sur l'exploration des zones où de bonnes solutions peuvent être trouvées.

--Même si elles ne semblent pas prometteuses au premier abord, la diversification vise à explorer différentes régions de l'espace de recherche. L'objectif est d'éviter de rester limité aux minimums locaux et d'explorer de nouvelles zones qui pourraient contenir de meilleures solutions. La diversification favorise l'exploration de nouvelles zones et la diversité des solutions. Cette méthode est comparable à l'exploration.

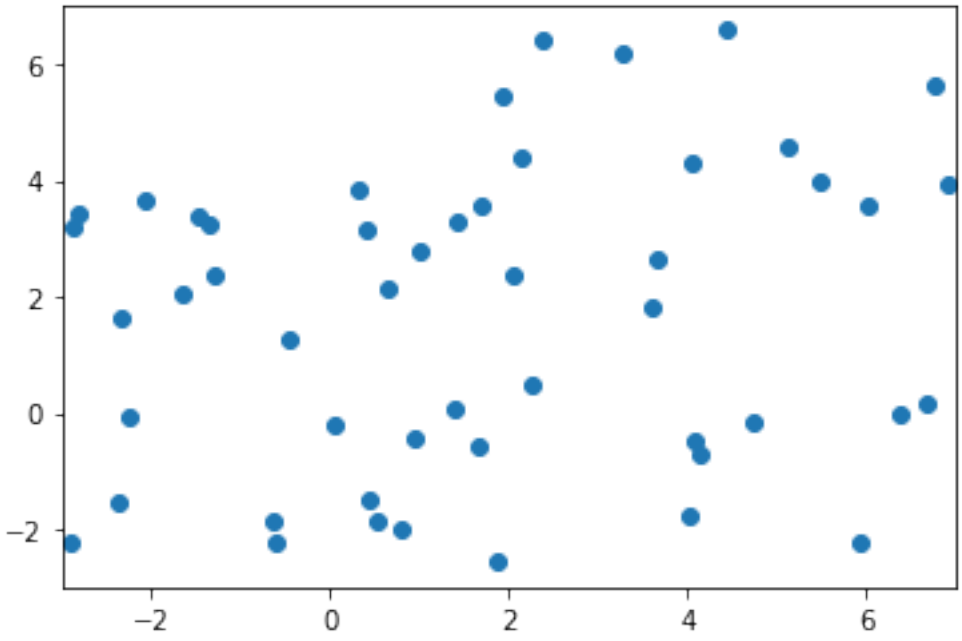
--EXEMPLE : L'algorithme génétique peut se concentrer sur les individus de la population ayant les meilleures valeurs d'adaptation (ou de fitness) pendant la phase d'intensification. Il peut exploiter ces individus et les améliorer progressivement en utilisant des opérations de sélection, de croisement et de mutation. Dans cette étape, les efforts peuvent être concentrés sur les domaines de l'espace de recherche où se trouvent les solutions de haute qualité.

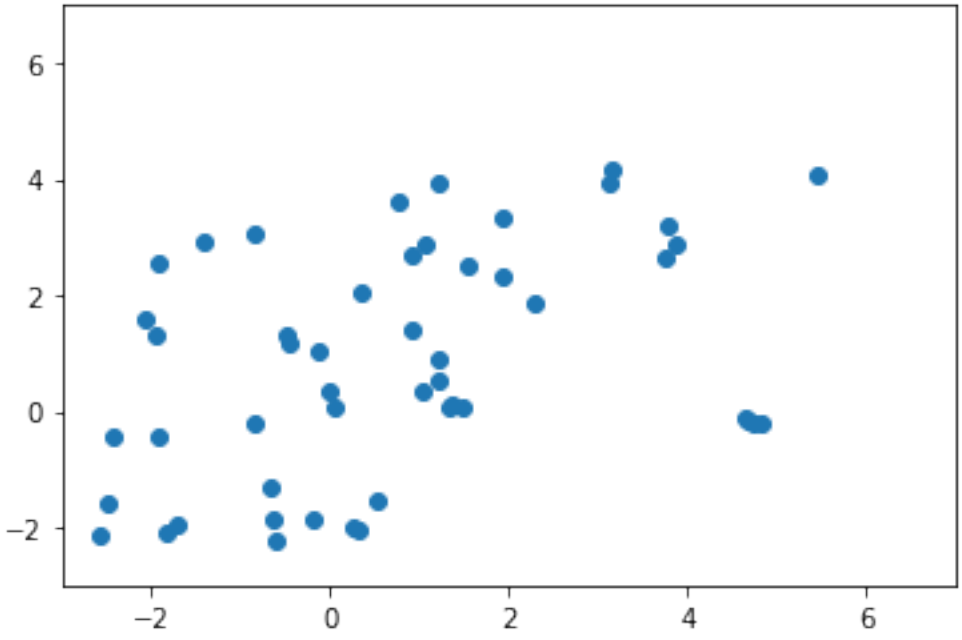
Pour encourager l'exploration de nouvelles régions de l'espace de recherche, l'algorithme génétique peut introduire des opérations de sélection, de croisement et de mutation plus exploratoires pendant la phase de diversification. Par exemple, il peut introduire des mécanismes de perturbation aléatoire dans la population ou utiliser des opérateurs de mutation plus agressifs pour permettre une exploration plus large de l'espace de recherche. Afin d'éviter une convergence prématurée vers des solutions locales et de trouver de nouvelles solutions potentiellement meilleures, cette étape est cruciale.

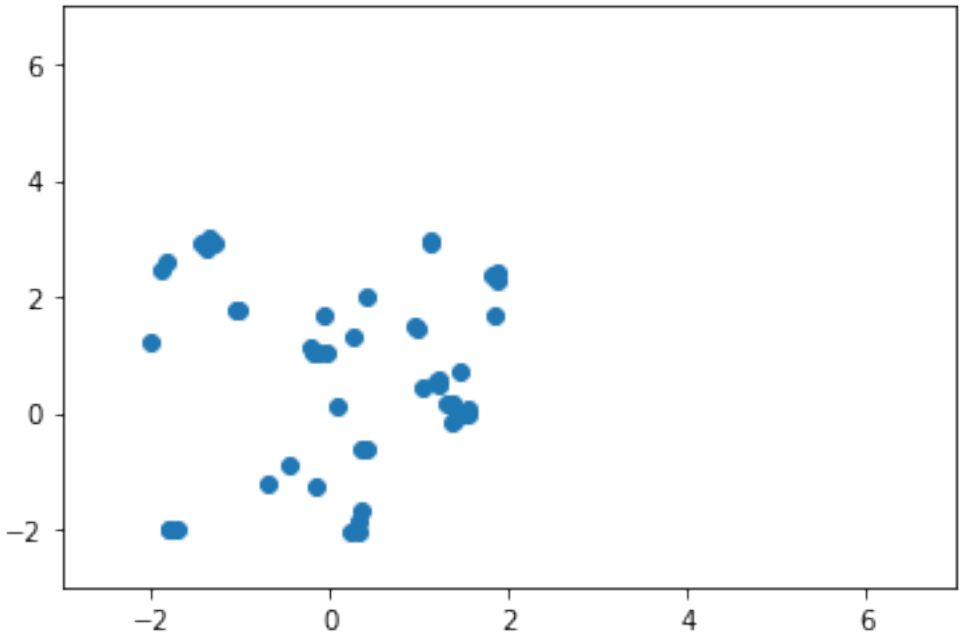
```
import numpy as np
```

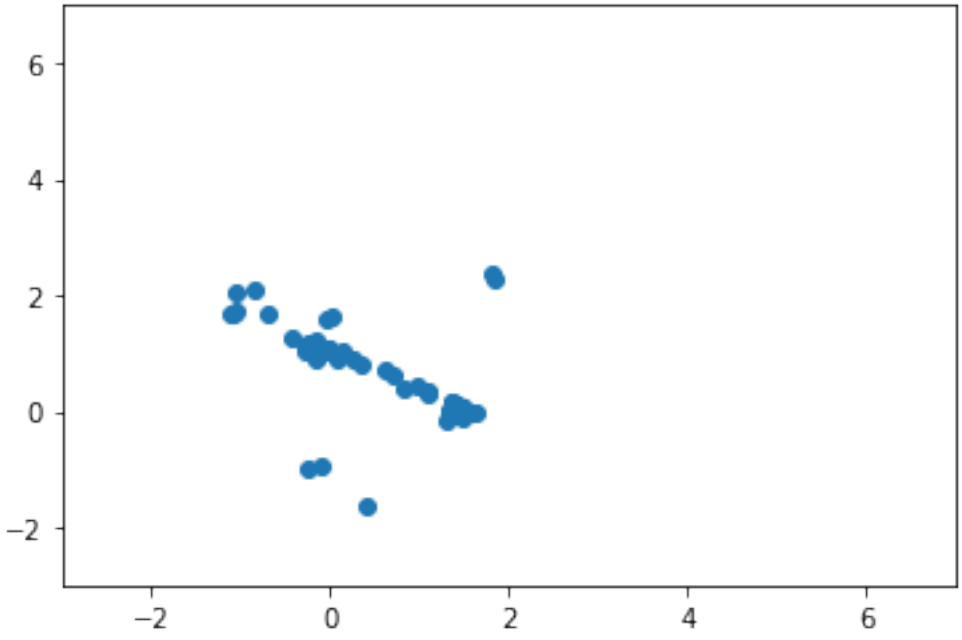
```
def rastrigin(x):
    n1 = len(x)
    y = n1 + np.sum(x**2 - np.cos(2 * np.pi * x))
    return y

def tournament_selection(A):
    Npop, n1 = A.shape
    Asel = np.copy(A)
    for i in range(Npop):
        u = np.random.randint(0, Npop)
        v = np.random.randint(0, Npop)
        if A[u, n1 - 1] < A[v, n1 - 1]:
            Asel[i, :] = A[u, :]
        else:
            Asel[i, :] = A[v, :]
    return Asel

def crossover(A, pc):
    Npop, n1 = A.shape
    Acrois = np.copy(A)
    for i in range(int(Npop / 2)):
        u = np.random.randint(0, Npop)
        v = np.random.randint(0, Npop)
        r = np.random.rand()
        if np.random.rand() < pc:
            Acrois[2 * i - 1, :] = r * A[u, :] + (1 - r) * A[v, :]
            Acrois[2 * i, :] = (1 - r) * A[u, :] + r * A[v, :]
        else:
            Acrois[2 * i - 1, :] = A[u, :]
            Acrois[2 * i, :] = A[v, :]
    return Acrois

def mutation(A, pm, alpha):
    Npop, n1 = A.shape
    Amut = np.copy(A)
    for i in range(Npop):
        if np.random.rand() < pm:
            for j in range(n1 - 1):
                Amut[i, j] = A[i, j] + 2 * alpha * (np.random.rand() -
0.5)
    return Amut

import matplotlib.pyplot as plt

Npop = 50
Ngen = 40
pm = 0.9
pc = 0.6
alpha = 0.1
n = 2
```

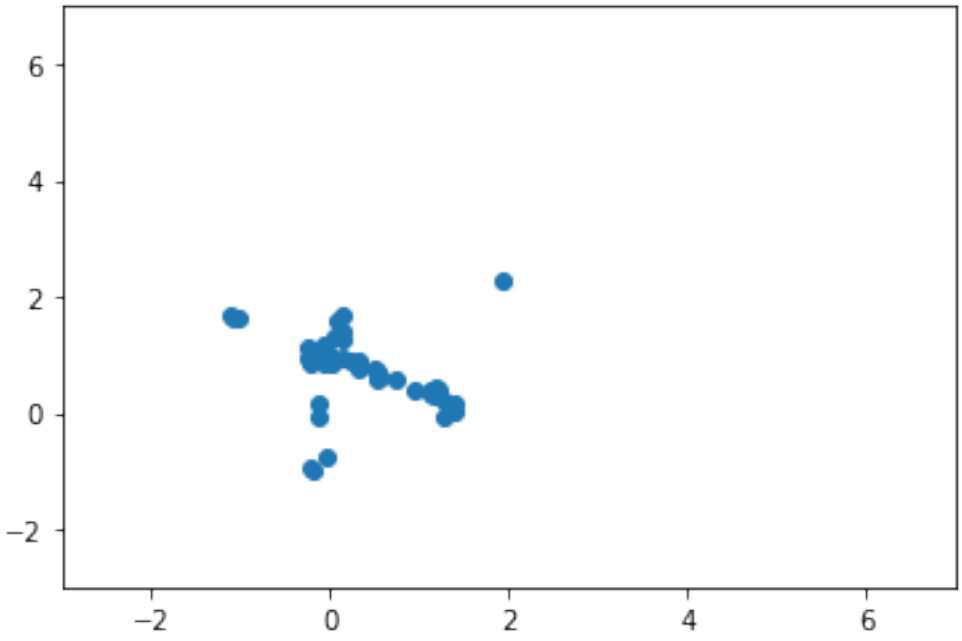
```
# Initialization
A = 10 * np.random.rand(Npop, n + 1) - 3 * np.ones((Npop, n + 1))

Jmin = []
plt.figure(1)

for i in range(Ngen):
    plt.clf()
    plt.plot(A[:, 0], A[:, 1], 'o')
    plt.axis([-3, 7, -3, 7])
    plt.draw()
    plt.pause(0.001)

    # Evaluation
    for j in range(Npop):
        A[j, n] = rastrigin(A[j, :n])

    c = np.argsort(A[:, n])

    Jmin.append(c[-1])

    # Selection
    Asel = tournament_selection(A)

    # Croisement
    Acrois = crossover(Asel, pc)

    # Mutation
    Amut = mutation(Acrois, pm, alpha)

    A = Amut

plt.figure(2)
plt.plot(range(Ngen), Jmin)
plt.xlabel('Generation')
plt.ylabel('Best Value (J)')
plt.show()
```

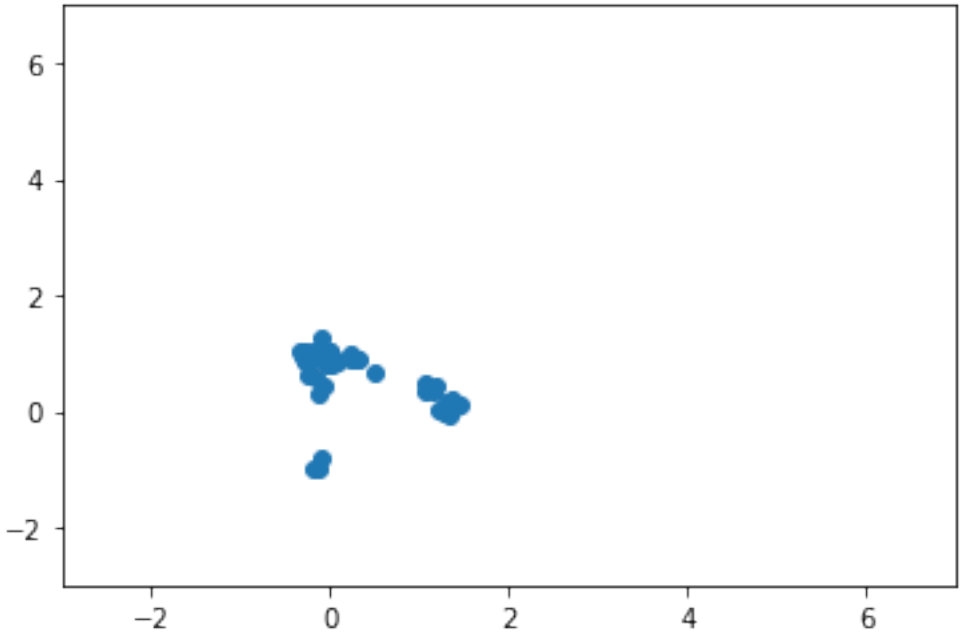

6
4
2
0
−2
−2
0
2
4
6

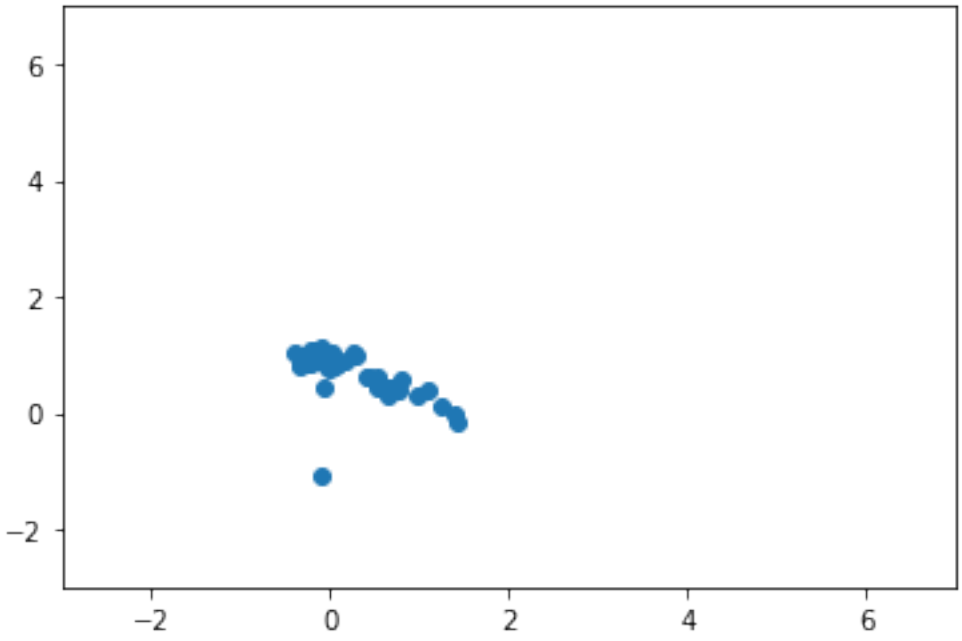

6
4
2
0
−2
−2
0
2
4
6

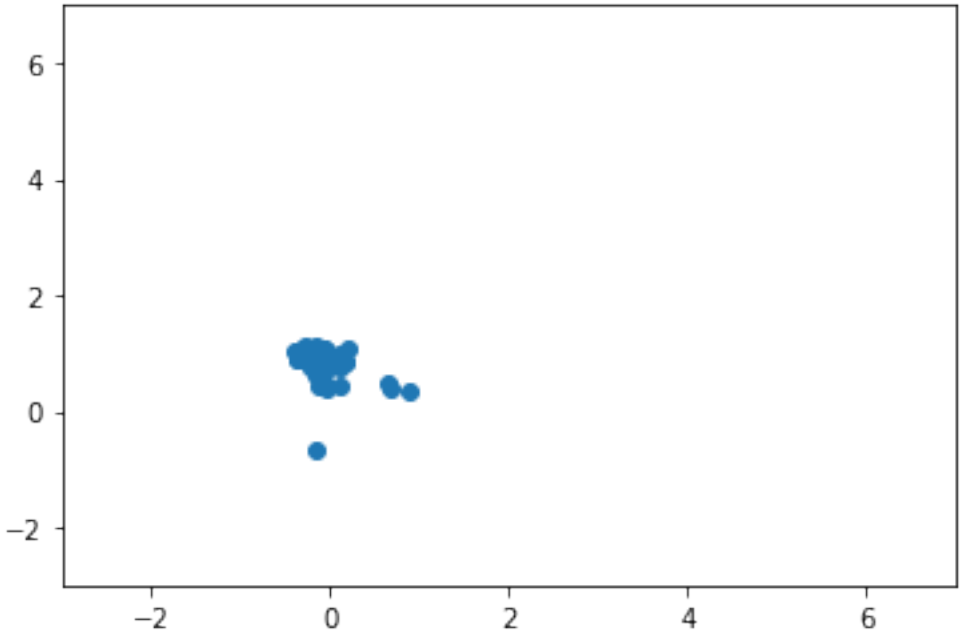

6
4
2
0
−2
−2
0
2
4
6

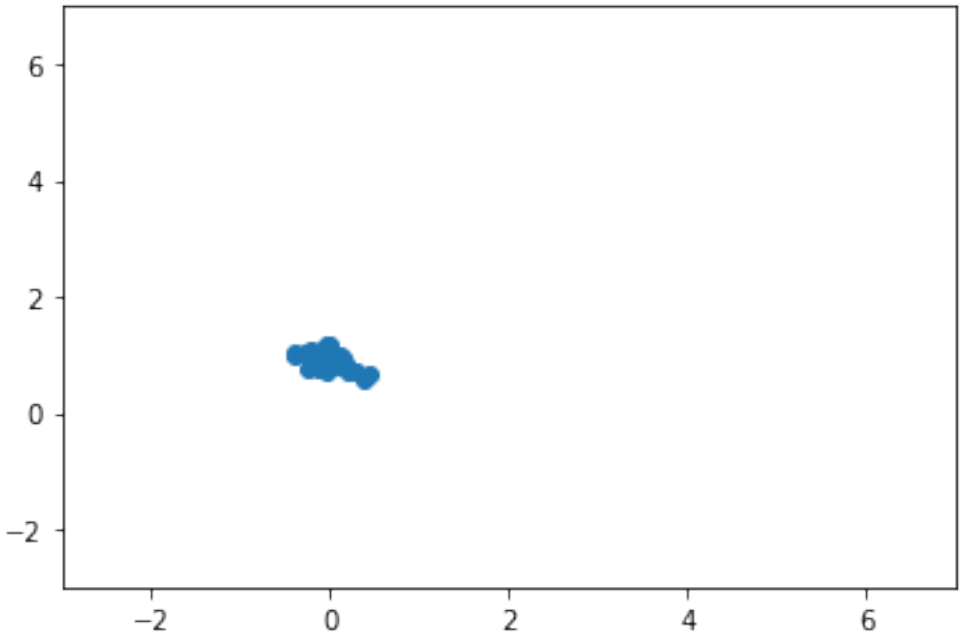

6
4
2
0
−2
−2
0
2
4
6

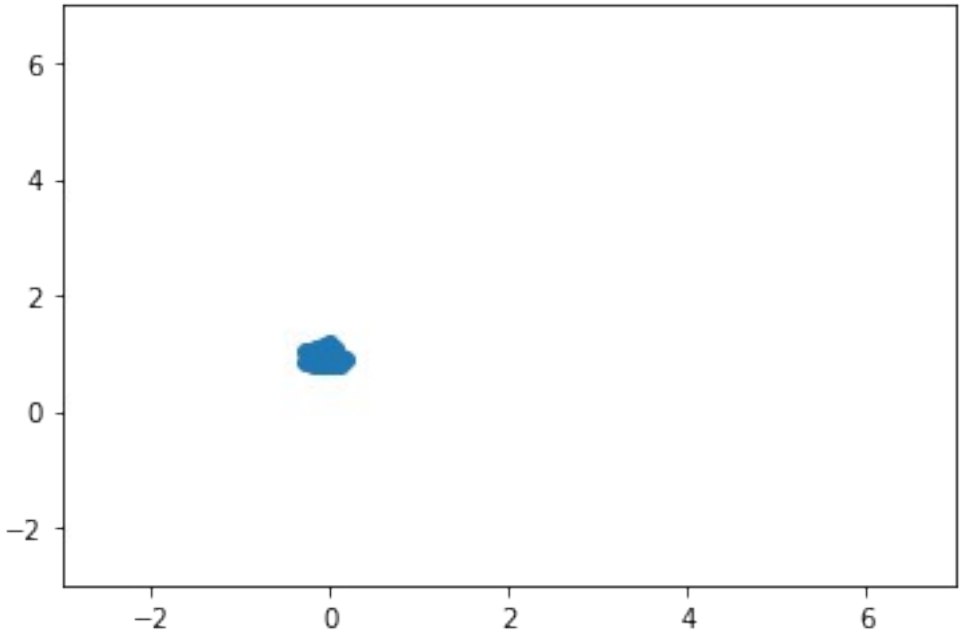

6
4
2
0
−2
−2
0
2
4
6

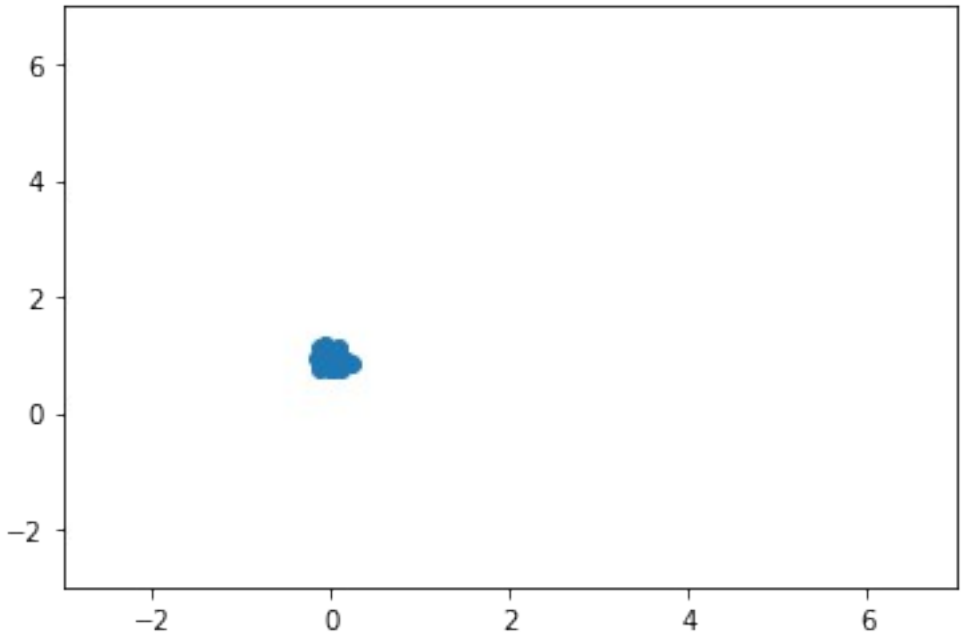

6
4
2
0
−2
−2
0
2
4
6

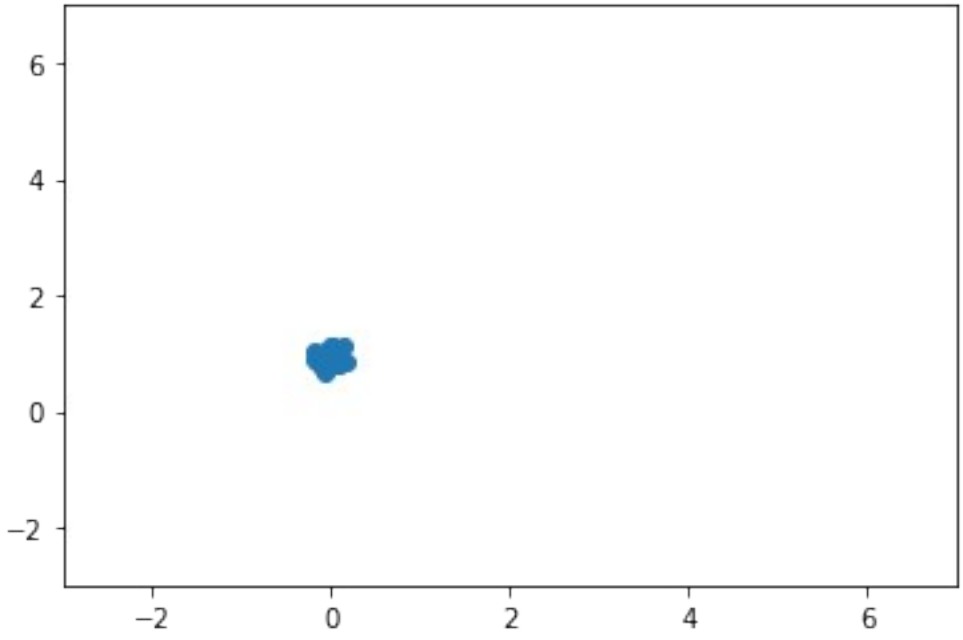

6
4
2
0
−2
−2
0
2
4
6

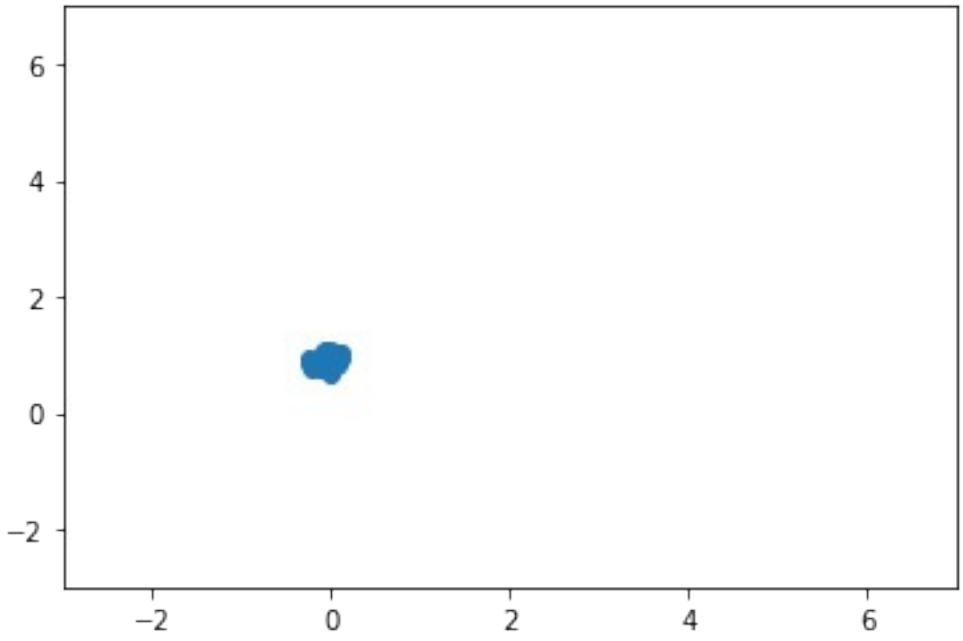

6
4
2
0
−2
−2
0
2
4
6

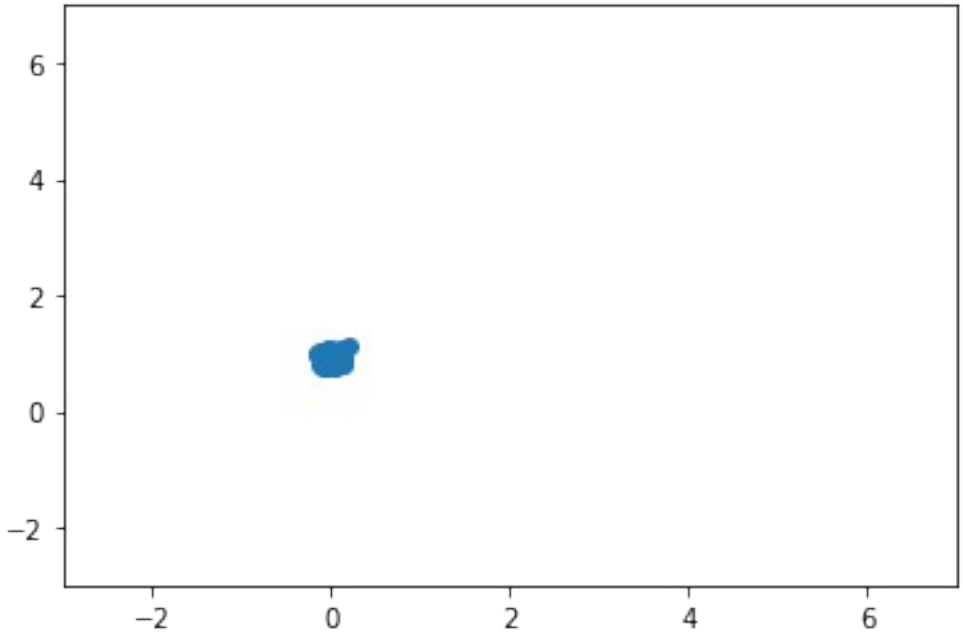

6
4
2
0
−2
−2
0
2
4
6

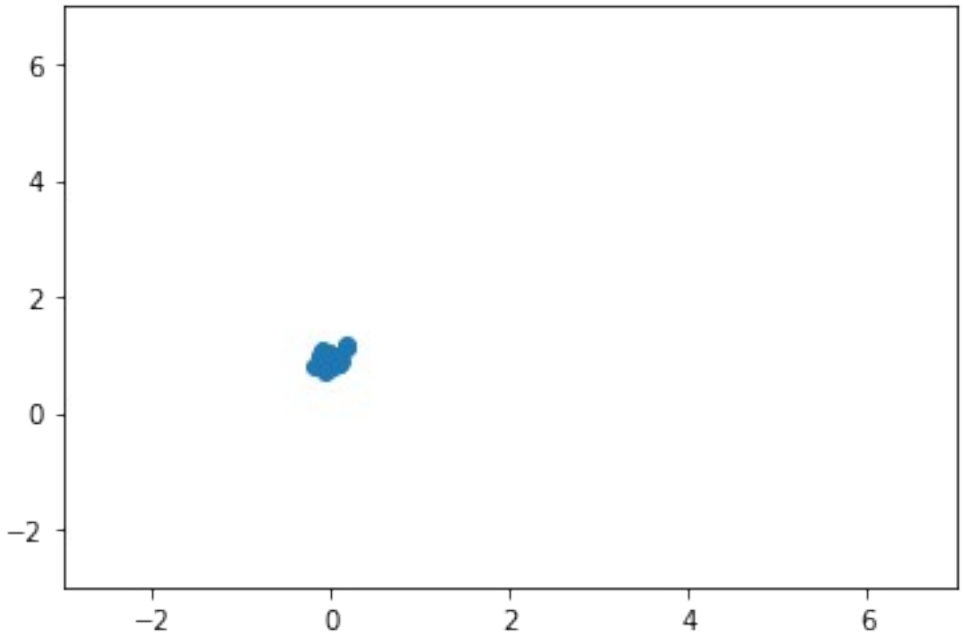

6
4
2
0
−2
−2
0
2
4
6

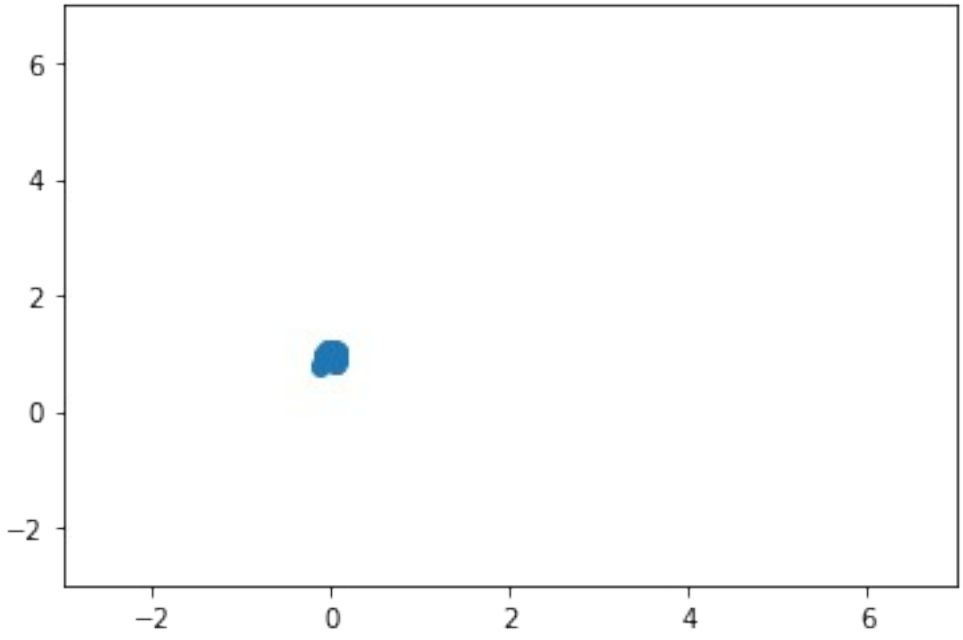

6
4
2
0
−2
−2
0
2
4
6

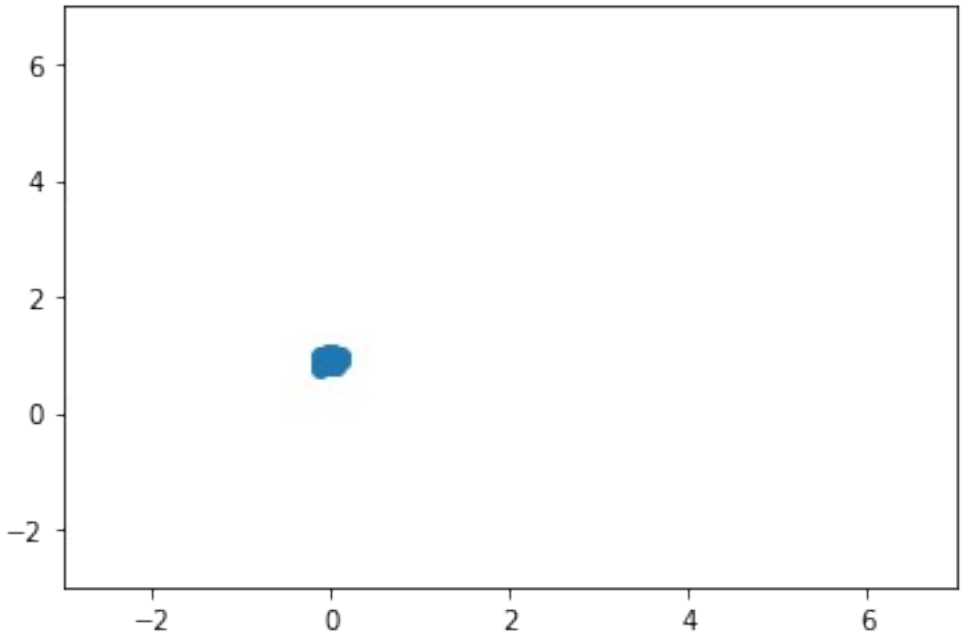

6
4
2
0
−2
−2
0
2
4
6

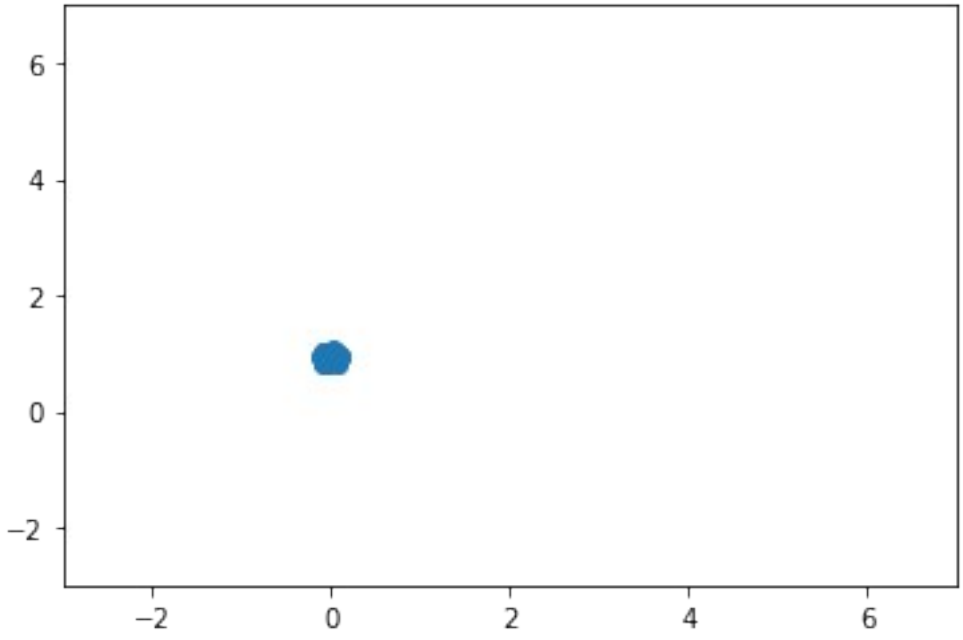

6
4
2
0
−2
−2
0
2
4
6

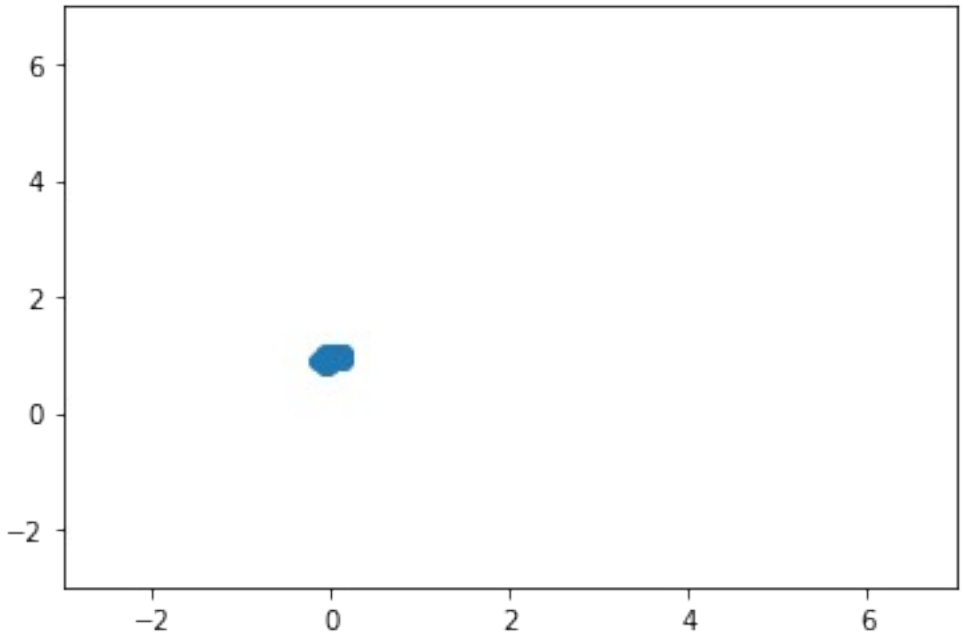

6
4
2
0
−2
−2
0
2
4
6

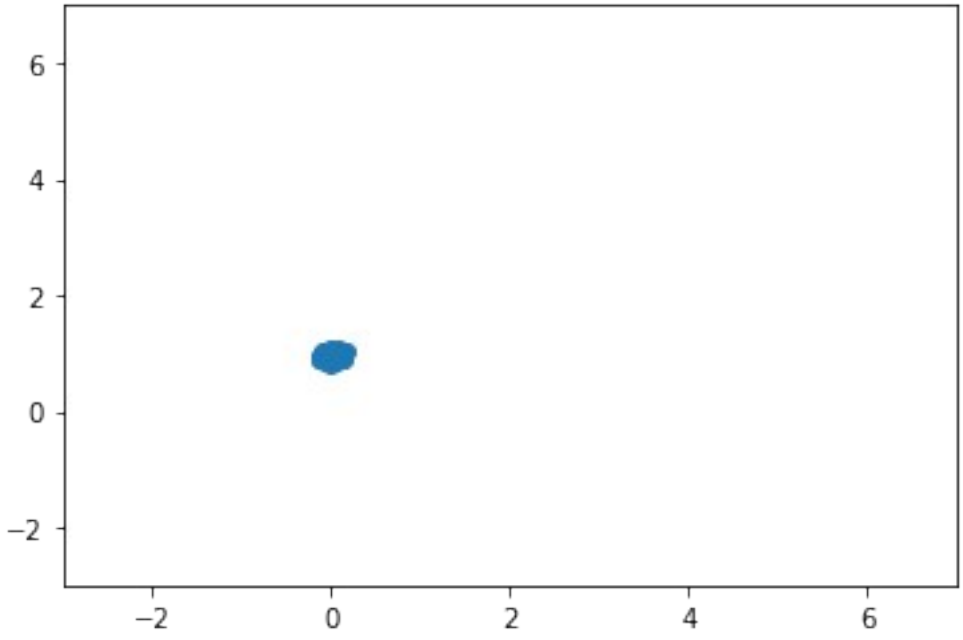

6
4
2
0
−2
−2
0
2
4
6

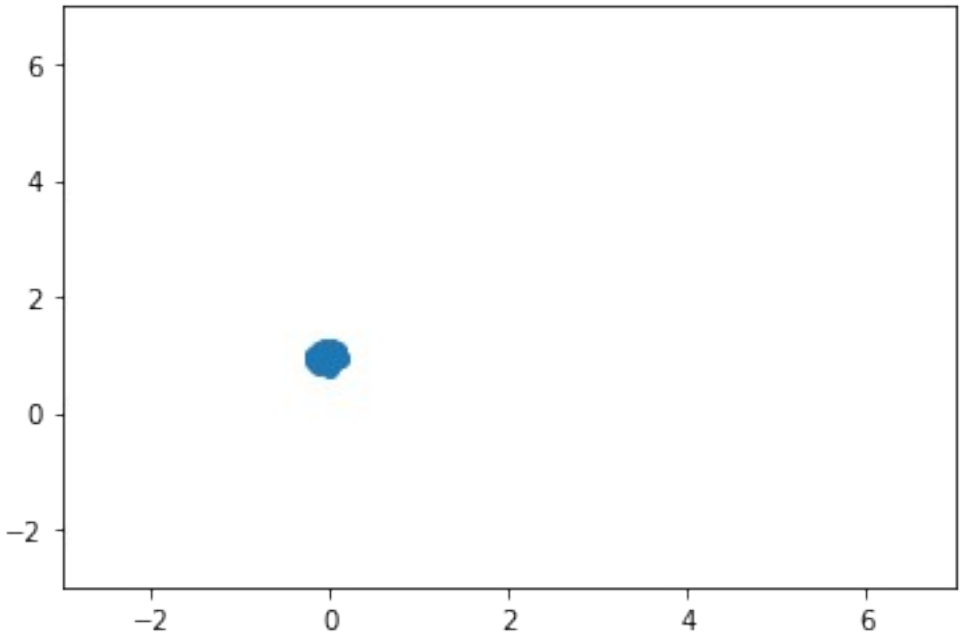

6
4
2
0
−2
−2
0
2
4
6

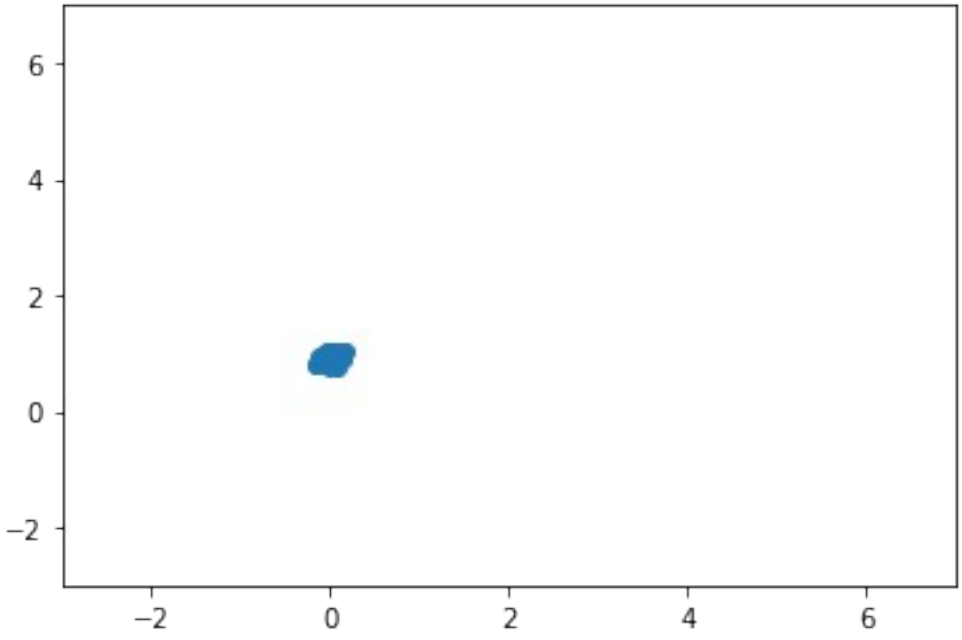

6
4
2
0
−2
−2
0
2
4
6

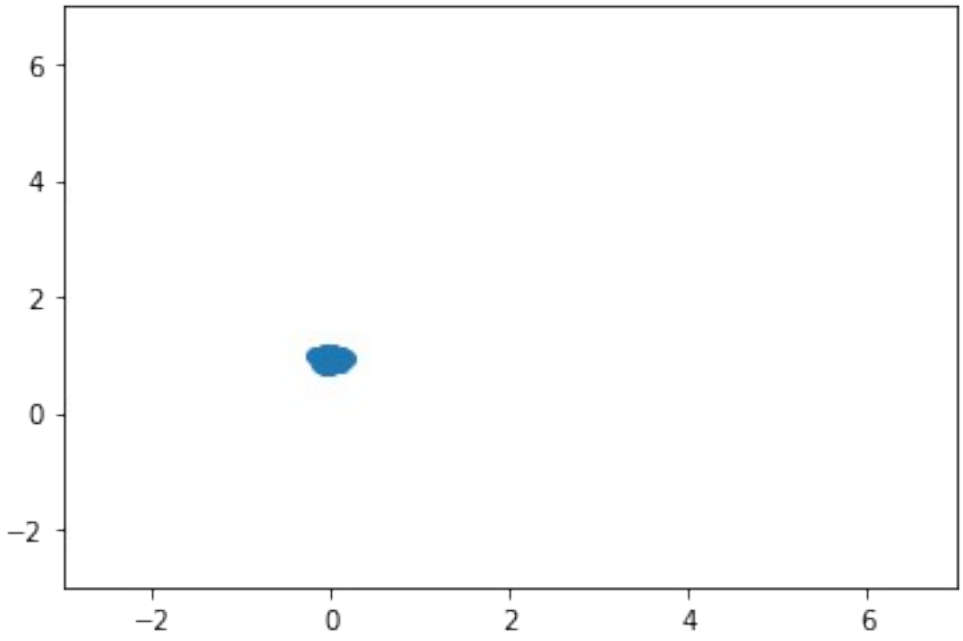

6
4
2
0
−2
−2
0
2
4
6

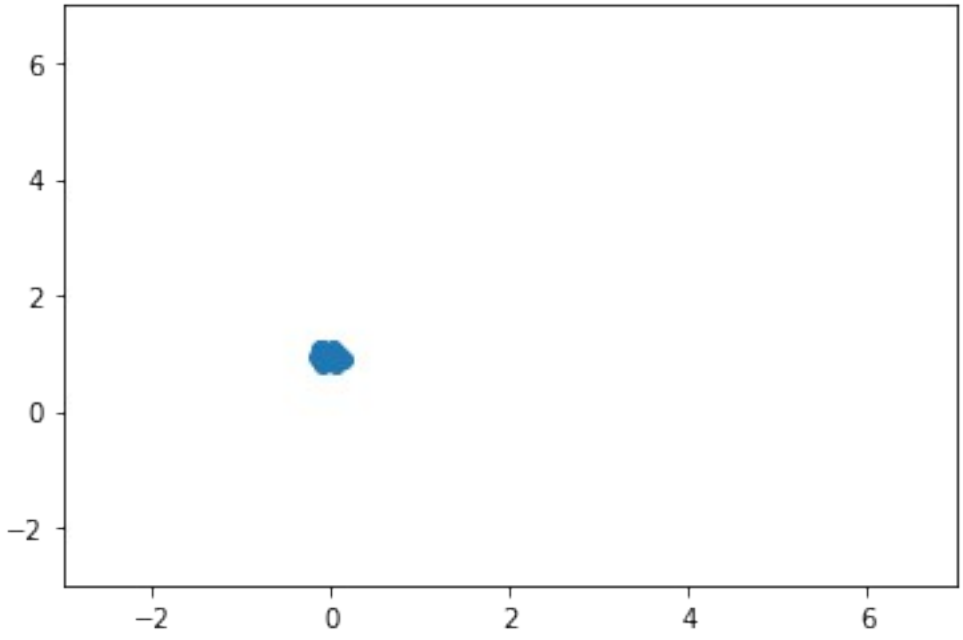

6
4
2
0
−2
−2
0
2
4
6

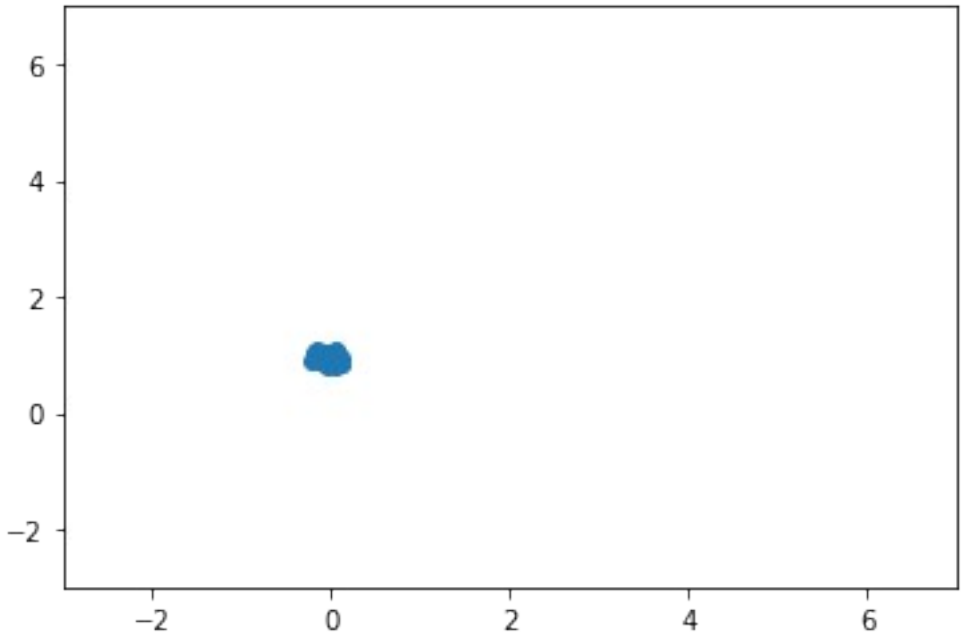

6
4
2
0
−2
−2
0
2
4
6

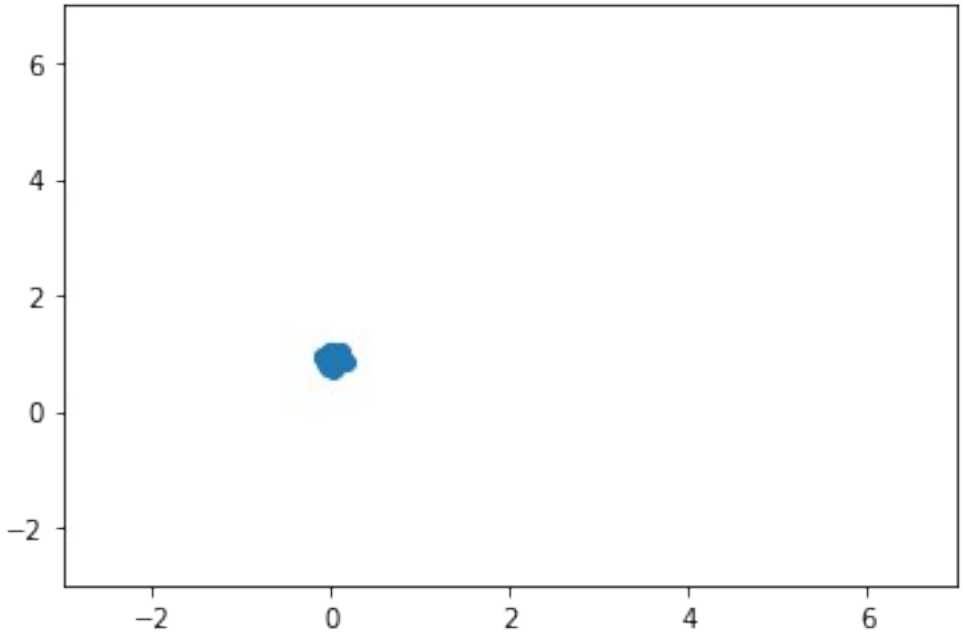

6
4
2
0
−2
−2
0
2
4
6

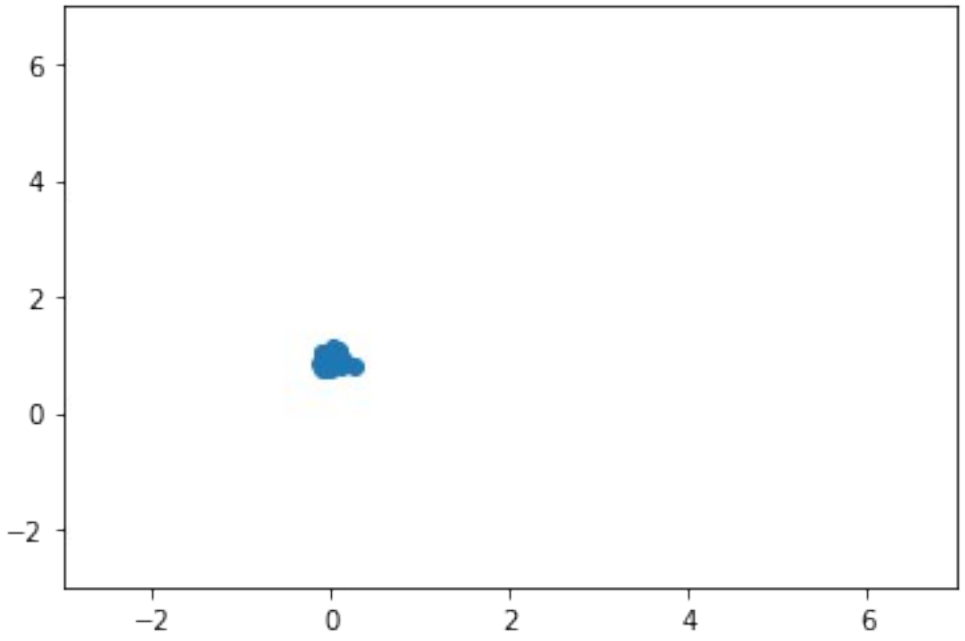

6
4
2
0
−2
−2
0
2
4
6

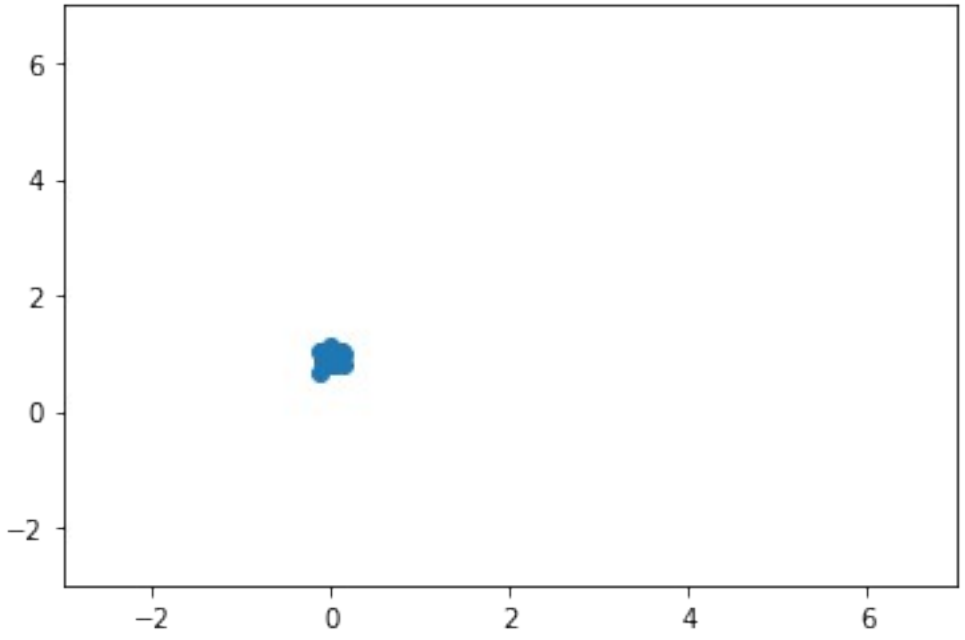

6
4
2
0
−2
−2
0
2
4
6

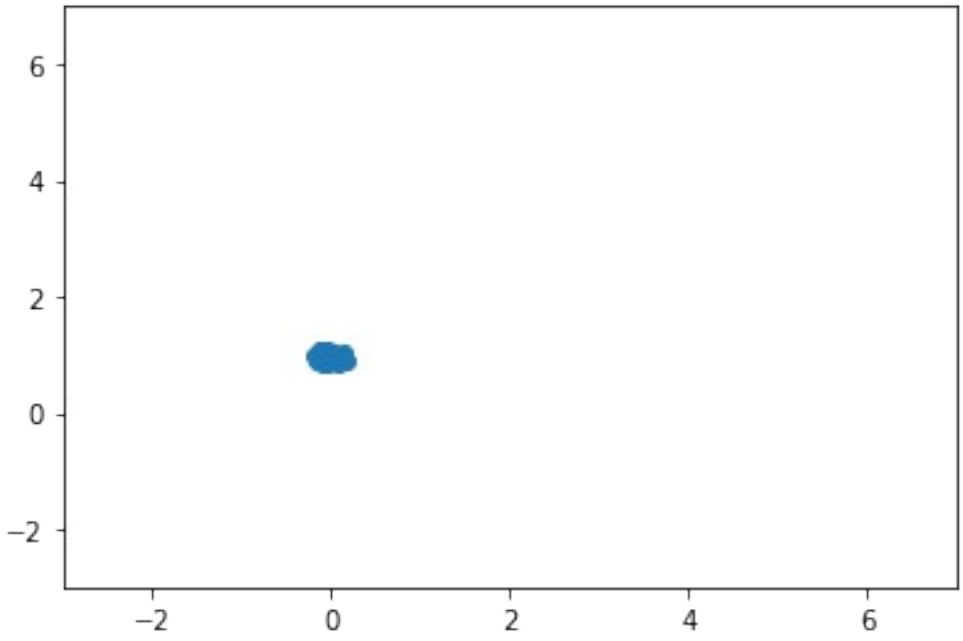

6
4
2
0
−2
−2
0
2
4
6

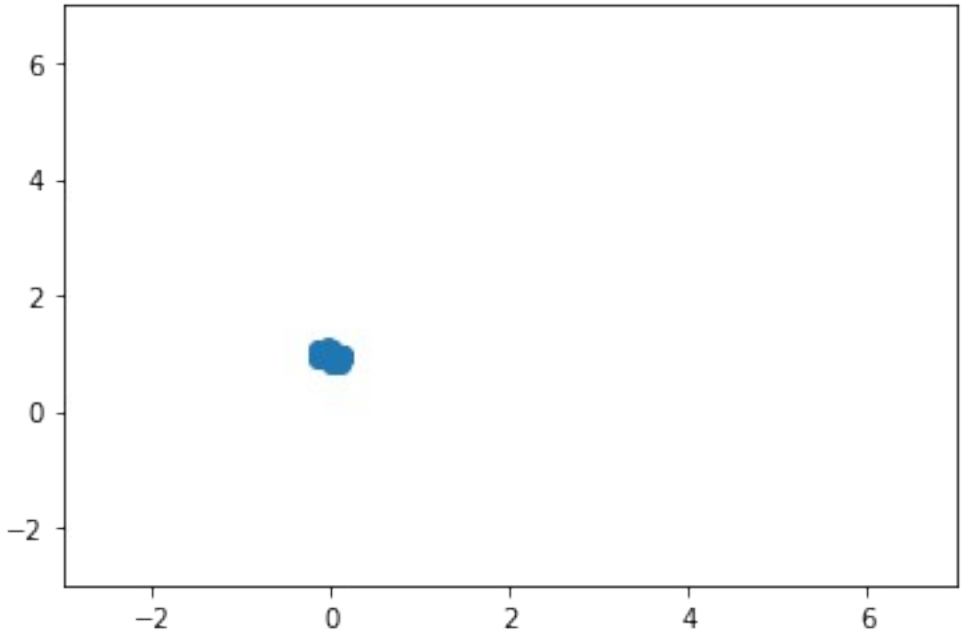

6
4
2
0
−2
−2
0
2
4
6

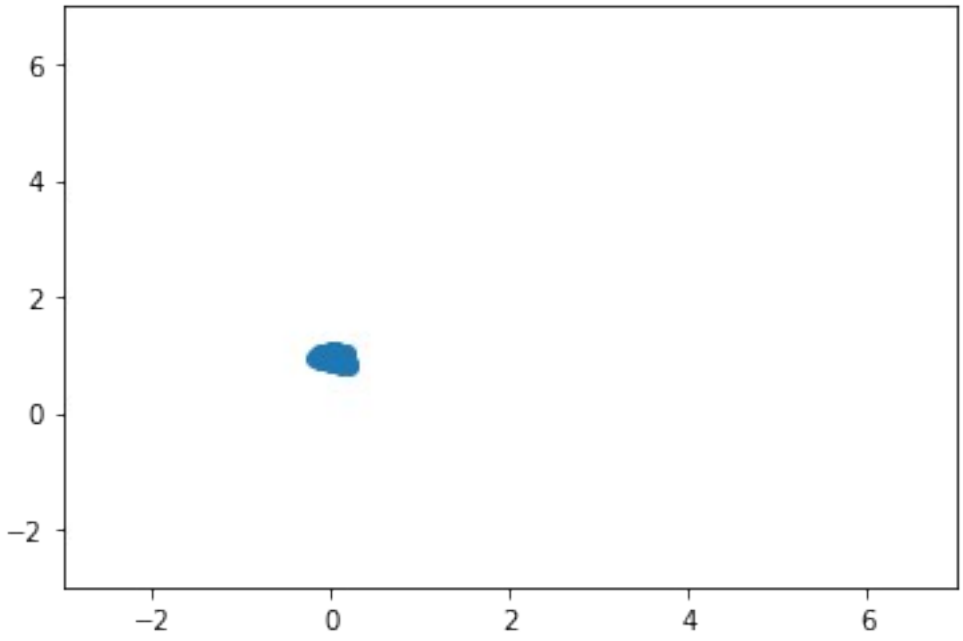

6
4
2
0
−2
−2
0
2
4
6

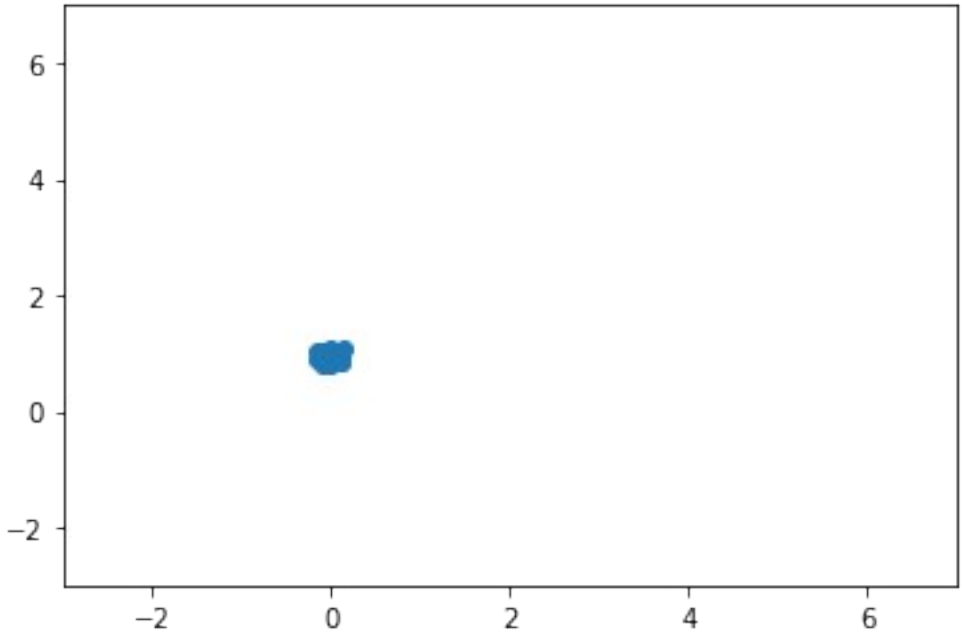

6
4
2
0
−2
−2
0
2
4
6

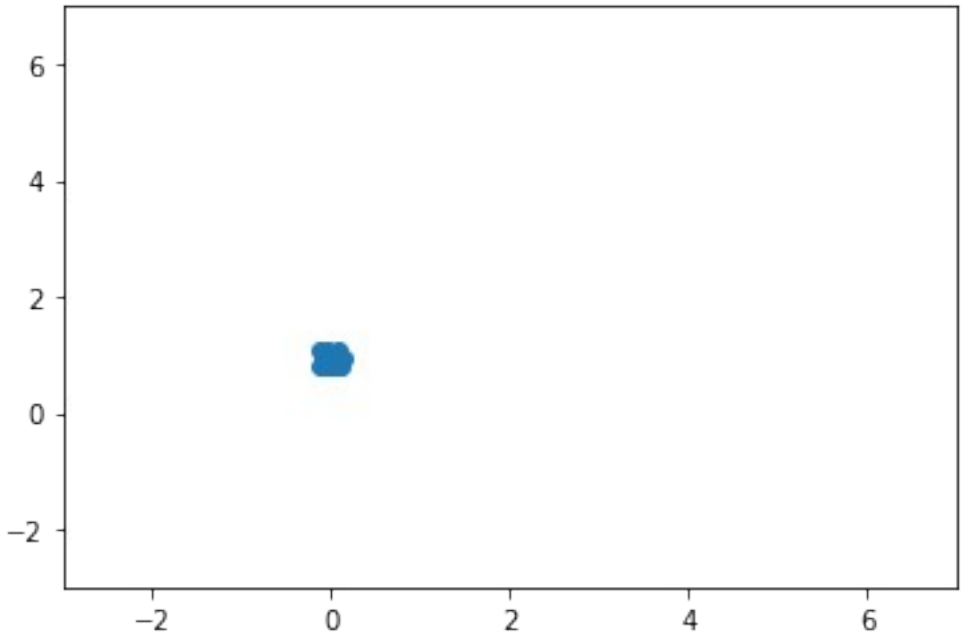

6
4
2
0
−2
−2
0
2
4
6

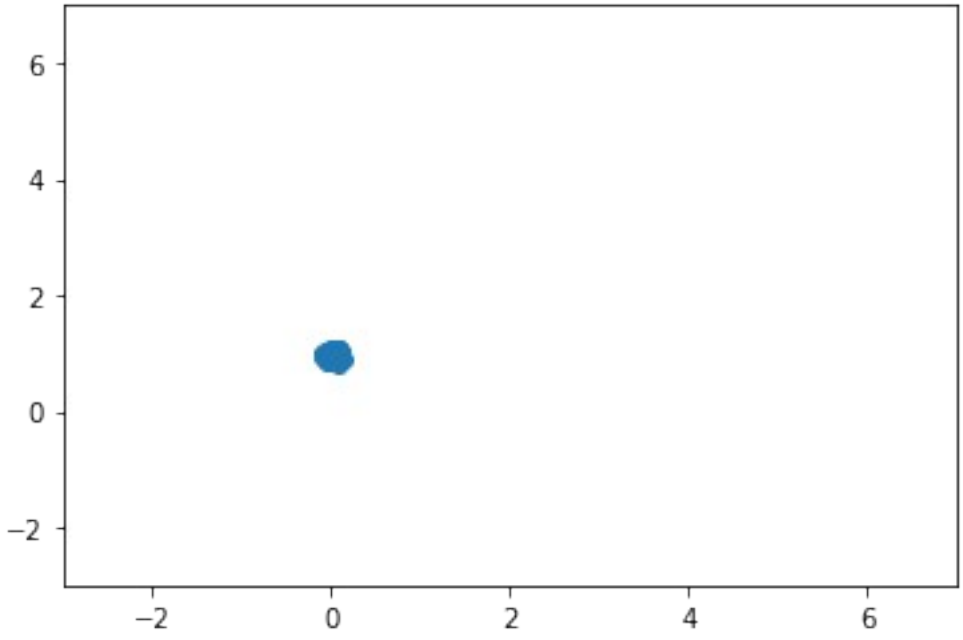

6
4
2
0
−2
−2
0
2
4
6

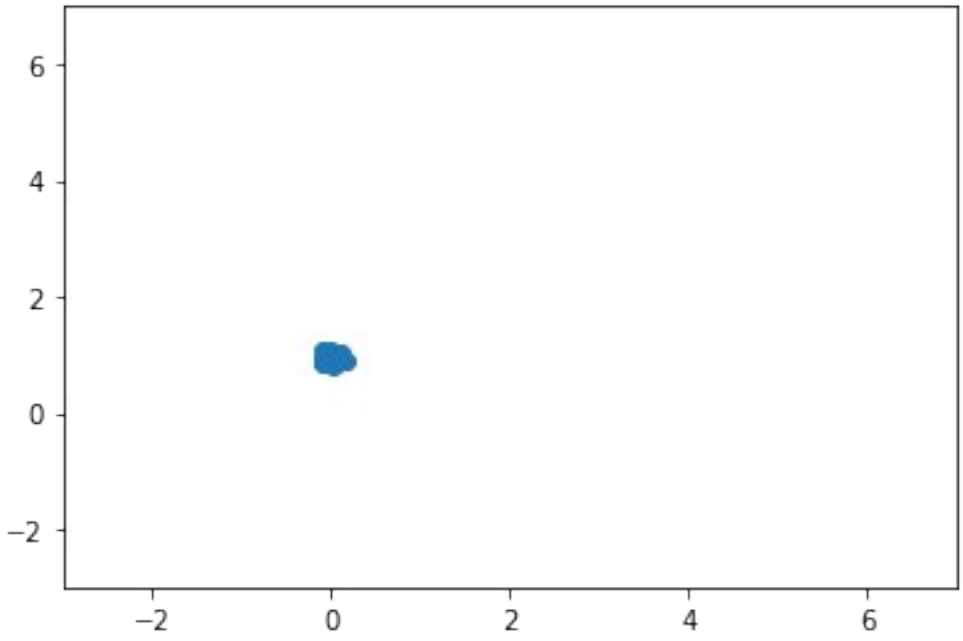

6
4
2
0
−2
−2
0
2
4
6

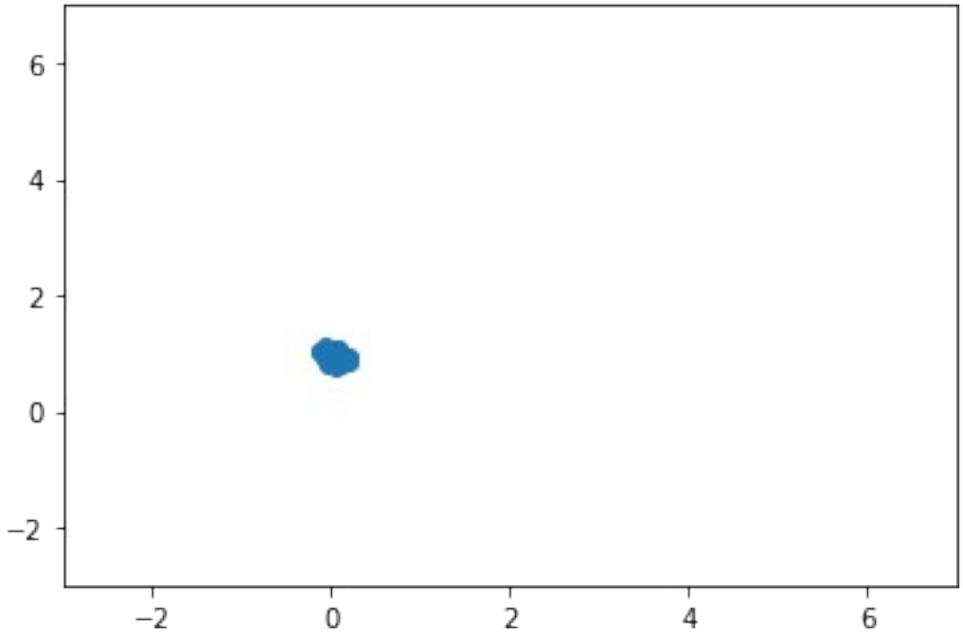

6
4
2
0
−2
−2
0
2
4
6

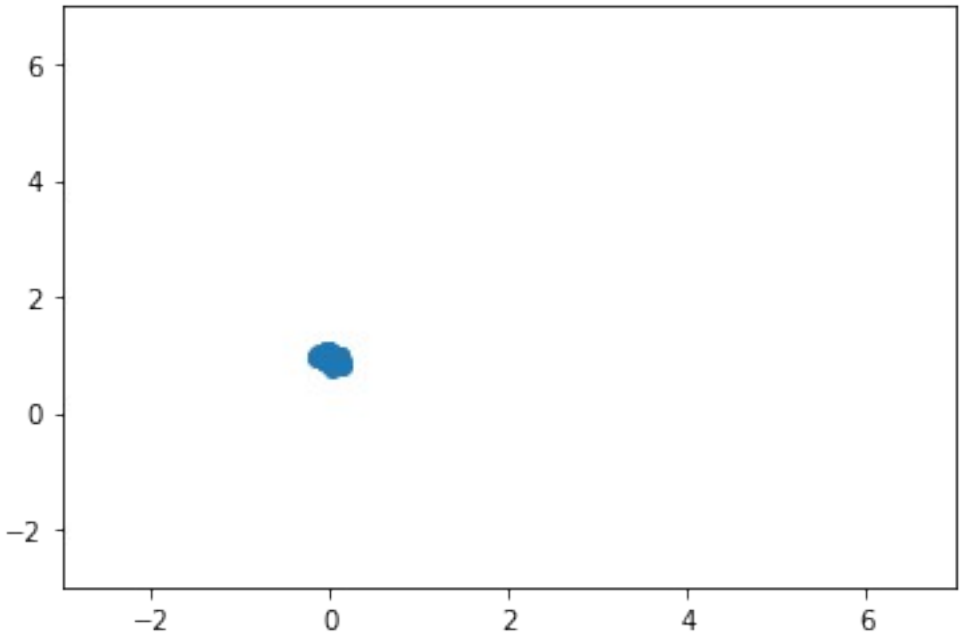

6
4
2
0
−2
−2
0
2
4
6

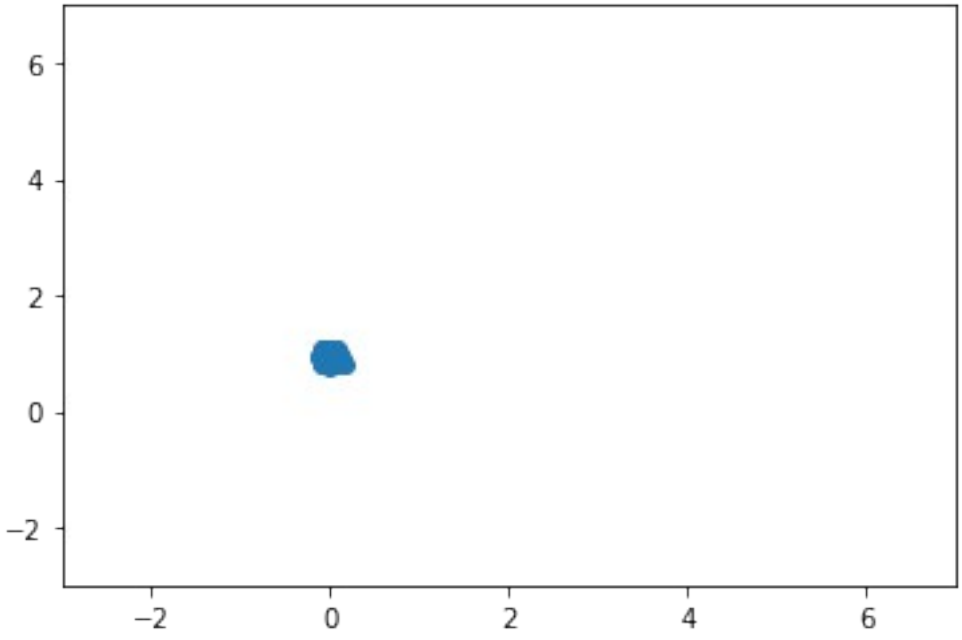

6
4
2
0
−2
−2
0
2
4
6

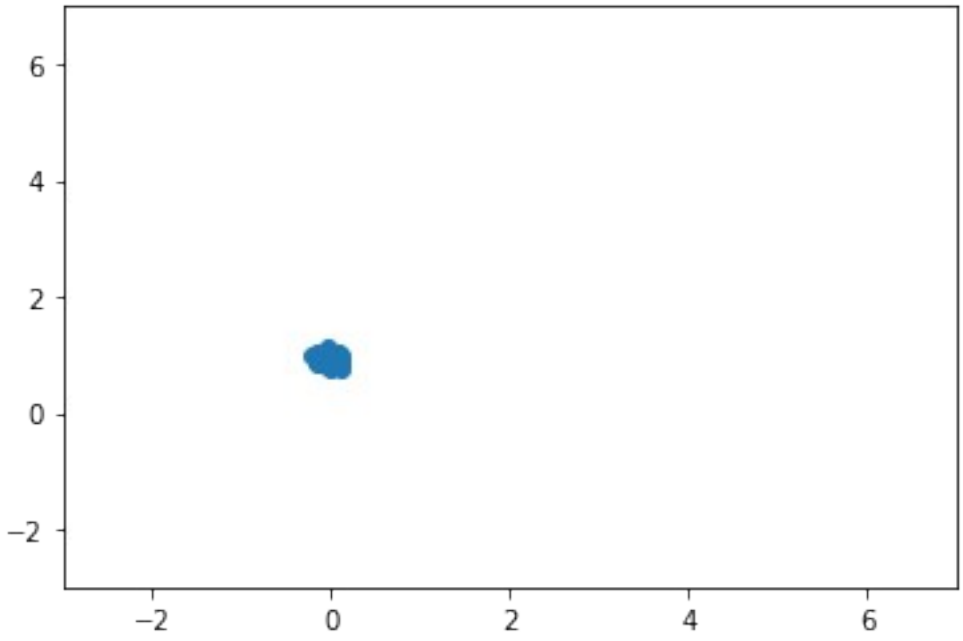

6
4
2
0
−2
−2
0
2
4
6

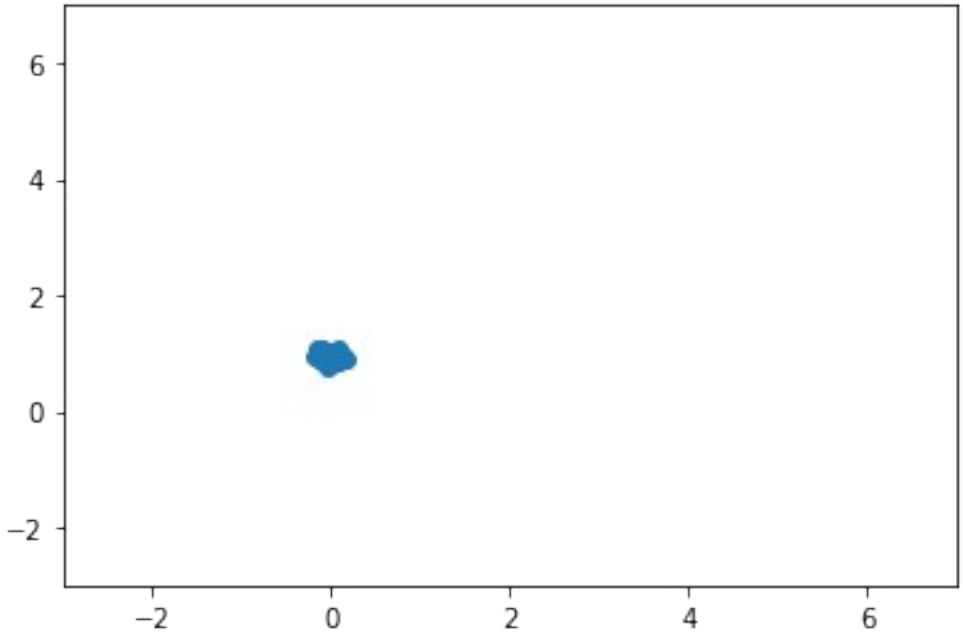

6
4
2
0
−2
−2
0
2
4
6

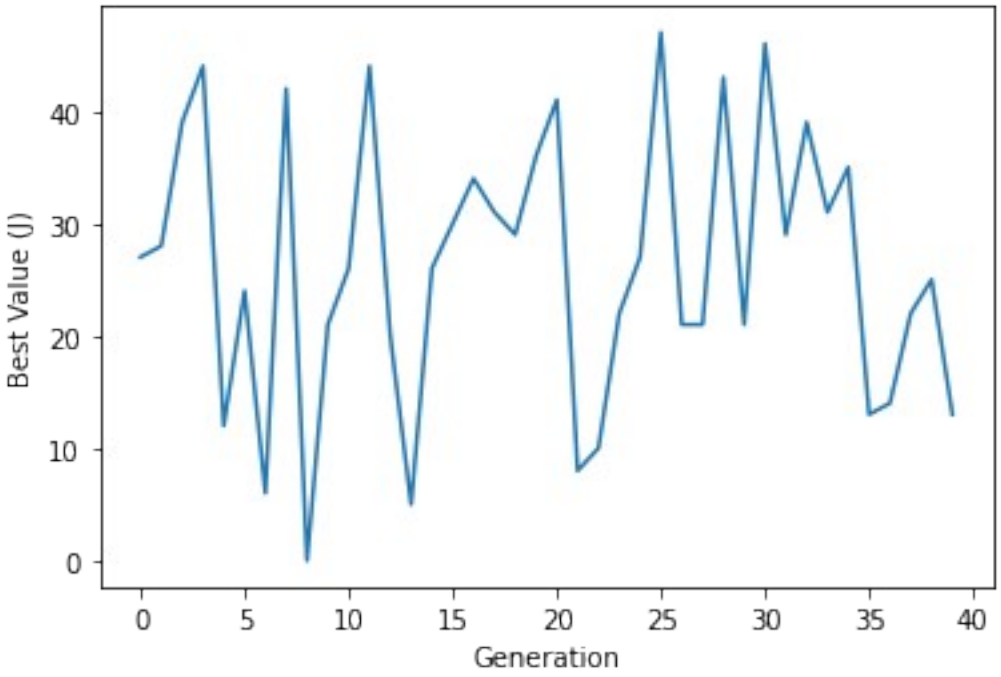

6
4
2
0
−2
−2
0
2
4
6


6
4
2
0
−2
−2
0
2
4
6


6
4
2
0
−2
−2
0
2
4
6


6
4
2
0
−2
−2
0
2
4
6


6
4
2
0
−2
−2
0
2
4
6


Best Value (J)
0
10
20
30
40
0
5
10
15
20
25
30
35
40
Generation