Simülasyon ve Programlama

Proje Ödevi

170420044 Ahmet Faruk Alkan

Boyutları a × b × c olan bir bölge içerisinde toplam N tane cisim vardır (a, b, c ve N kullanıcı parametreleridir).

1. (a) a, b, c ve N parametrelerini elde ediniz.

   (b) N tane cismi bu bölge içerisinde rastgele olarak dağıtınız.

   (c) Sistemin anlık görüntüsünü elde ediniz.

   (d) Cisimler arasındaki ortalama mesafeyi elde ediniz.

   (e) Cisimlerden birini rastgele seçiniz, bu cismin konumunu rastgele değiştiriniz (3D olacak şekilde). (Burada maksimum yer değiştirme miktarını tanımlayınız). Cisimlerden biri, bu bölgenin bir kenarından dışarı çıkması durumunda cismin diğer kenardan içeri giriyor olduğunu sağlayınız (bakınız ödev2).

   (f) Bir önceki adımı T defa tekrarlayınız. Burada T simülasyon adım miktarı olup girdi parametresi olmasını sağlayınız.

   (g) Cisimler arasındaki ortalama mesafenin zaman ile değişimini görüntüleyiniz.

   (h) Sistemin dengeye gelip gelmediğini kontrol ediniz. Simülasyonun başından itibaren örneğin cisimler arasındaki mesafelerin ortalamasına bakınız, burada dalgalanmaların azaldığı anı dengeye ulaşma anı olarak tanımlayabilirsiniz.

   (i) Denge durumları üzerinden cisimler arasındaki ortalama mesafeyi okuyunuz.

   (j) Cisimler arasındaki mesafe için bir histogram yapınız.

   (k) Sistemin anlık görüntüsünü elde ediniz.

   Yukarıdaki simülasyonda konumu değiştirilen cisimin yeni konumuna herhangi bir kabul kriteri olmadan taşındığını not ediniz. Yani simülasyonda yeni konumların kabul edilme olasılığı %100'dür.

Açıklama:

calculateDistance: İki nokta arasındaki mesafeyi hesaplayan bir fonksiyondur.

generateObjects: N tane cismin rastgele olarak (0-a, 0-b, 0-c) aralığında konumlandırıldığı bir liste oluşturan bir fonksiyondur.

moveObject: Belirli bir cismin konumunu rastgele olarak değiştiren ve yeni konumunu döndüren bir fonksiyondur. Yer değiştirme miktarı maxDisplacement ile sınırlıdır ve cisim, bölgenin bir kenarından çıktığında diğer kenardan içeri girer.

calculateAvgDistance: Tüm cisimler arasındaki ortalama mesafeyi hesaplayan bir fonksiyondur.

simulate: Ana simülasyon fonksiyonudur. Önce cisimler oluşturulur, ardından T adımı boyunca bir cisim seçilir, yer değiştirilir ve cisimler arasındaki ortalama mesafe kaydedilir.

threeD: Cisimlerin 3D görüntüsünü matplotlib kullanarak çizen bir fonksiyondur.

distanceGraghich: Cisimler arasındaki ortalama mesafenin zamanla değişimini grafiğe döken bir fonksiyondur.

main: Kullanıcıdan parametreleri alır, simülasyonu çalıştırır ve sonuçları görüntüler.

Çıkarımlar:

Cisimler Arasındaki Ortalama Mesafe: Simülasyon boyunca cisimler arasındaki ortalama mesafe kaydedilmiştir. Bu mesafe, cisimlerin birbirlerine olan yakınlığını veya uzaklığını temsil eder. Simülasyonun ilerleyen adımlarında ortalama mesafenin değiştiği görülür. Bu, cisimlerin konumlarının zamanla farklılaştığını ve birbirlerinden uzaklaştığını veya yakınlaştığını gösterir.

Denge Durumu: Cisimler arasındaki ortalama mesafenin zamanla değişimine bakarak, dengeye ulaşıp ulaşmadığını kontrol ederiz. Eğer ortalama mesafe zamanla sabitlenir ve dalgalanmalar azalırsa, sistemin dengeye geldiği söylenebilir. Bu, cisimlerin konumlarının istikrarlı hale geldiğini ve bir denge durumuna ulaştığını gösterir.

Kod:

```
import random
import matplotlib.pyplot as plt

"""
Boyutları a × b × c olan bir bölge içerisinde toplam N tane cisim vardır
(a, b, c ve N kullanıcı parametreleridir).
1. (a) a, b, c ve N parametrelerini elde ediniz.
(b) N tane cismi bu bölge içerisinde rastgele olarak dağıtınız.
(c) Sistemin anlık görüntüsünü elde ediniz.
(d) Cisimler arasındaki ortalama mesafeyi elde ediniz.
(e) Cisimlerden birini rastgele seçiniz, bu cismin konumunu rastgele
değiştiriniz (3D olacak şekilde). (Burada maksimum yer değiştirme miktarını
tanımlayınız). Cisimlerden biri, bu bölgenin bir kenarından dışarı çıkması
durumunda cismin diğer kenardan içeri giriyor olduğunu sağlayınız (bakınız
ödev2).
(f) Bir önceki adımı T defa tekrarlayınız. Burada T simülasyon adım miktarı
olup girdi parametresi olmasını sağlayınız.
(g) Cisimler arasındaki ortalama mesafenin zaman ile değişimini
görüntüleyiniz.
(h) Sistemin dengeye gelip gelmediğini kontrol ediniz. Simülasyonun
başından itibaren örneğin cisimler arasındaki mesafelerin ortalamasına
bakınız, burada dalgalanmaların azaldığı anı dengeye ulaşma anı olarak
tanımlayabilirsiniz.
```

(i) Denge durumları üzerinden cisimler arasındaki ortalama mesafeyi okuyunuz.
(j) Cisimler arasındaki mesafe için bir histogram yapınız.
(k) Sistemin anlık görüntüsünü elde ediniz.

Yukarıdaki simülasyonda konumu değiştirilen cisimin yeni konumuna herhangi bir kabul kriteri olmadan taşındığını not ediniz. Yani simülasyonda yeni konumların kabul edilme olasılığı %100'dür.

"""

```
def calculateDistance(p1, p2):
    return ((p2[0] - p1[0]) ** 2 + (p2[1] - p1[1]) ** 2 + (p2[2] - p1[2])
** 2) ** 0.5

def generateObjects():
    objects = []
    for i in range(N):
        x = random.uniform(0, a)
        y = random.uniform(0, b)
        z = random.uniform(0, c)
        objects.append((x, y, z))
    return objects

def moveObject(obj):
    x, y, z = obj
    new_x = random.uniform(max(x - maxDisplacement, 0), min(x +
maxDisplacement, a))
    new_y = random.uniform(max(y - maxDisplacement, 0), min(y +
maxDisplacement, b))
    new_z = random.uniform(max(z - maxDisplacement, 0), min(z +
maxDisplacement, c))
    return new_x, new_y, new_z

def calculateAvgDistance():
    total_distance = 0
    counter = 0
    for i in range(len(objects)):
        for j in range(i + 1, len(objects)):

            distance = calculateDistance(objects[i], objects[j])
            total_distance += distance
            counter += 1

    return total_distance / counter

def simulate():
    global objects, distances
    objects = generateObjects()
    distances = []

    for i in range(T):
        randomObject = random.choice(objects)
        objects.remove(randomObject)
        objects.append(moveObject(randomObject))
        distances.append(calculateAvgDistance())

def threeD():
    fig = plt.figure()
    ax = fig.add_subplot(111, projection='3d')
```

```
    xs = [obj[0] for obj in objects]
    ys = [obj[1] for obj in objects]
    zs = [obj[2] for obj in objects]

    ax.scatter(xs, ys, zs)

    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.set_zlabel('Z')

    plt.show()


def distanceGraghich():
    plt.plot(range(T), distances)
    plt.xlabel('Adım Sayısı')
    plt.ylabel('Ortalama Mesafe')
    plt.show()

def main():
    global a, b, c, N, maxDisplacement, T
    print("3 boyutlu evrenin büyüklük değerleri:")
    a = float(input("a: "))
    b = float(input("b: "))
    c = float(input("c: "))
    N = int(input("Kaç tane cisim olsun: "))
    maxDisplacement = float(input("Maksimum yer değiştirme miktarı: "))
    T = int(input("Simülasyon adım sayısı: "))
    simulate()
    threeD()
    distanceGraghich()



main()
```

2. Yukarıdaki simülasyonu yeniden değerlendirelim. 1. soruda seçilen cisme rastgele yer değiştirme yaptırmak istediğimizde, geçmek istediği konum yakınında dlim kadar ötede başka bir cismin olup olmadığını kontrol edelim. Burada dlim bir parametredir. Eğer bu uzaklık içerisinde herhangi bir başka cisim yoksa geçiş tamamlanır. Eğer bu uzaklık içerisinde başka bir cisim varsa geçiş %50 olasılıkla tamamlanır. Bunu gerçekleştirmek için, geçilmek istenen konumdaki en yakın diğer cismin uzaklığı hesaplanır. Örneğin dx olsun ve eğer e^(-dx) ≥ R ise geçiş sağlanır. Aksi halde sağlanmaz. Burada R, [0, 1] aralığında rastgele bir sayıdır.

   (a) Sistemin dengeye gelip gelmediğini kontrol ediniz.

   (b) Cisimler arasındaki ortalama mesafenin zaman ile değişimine bakınız.

   (c) Denge durumları üzerinden cisimler arasındaki ortalama mesafeyi bulunuz.

   (d) Sistemin denge durumuna ait anlık görüntüler elde ediniz.

   (e) Yeni durumlara geçiş oranını elde ediniz ve görüntüleyiniz. Geçiş oranını yaklaşık %65 civarında olması için neler yapabileceğinizi değerlendiriniz, yani çok küçük veya çok yüksek değerler olmaması için.

Açıklama:

Kullanıcıdan evrenin boyutları (a, b, c), cisim sayısı (N), maksimum yer değiştirme miktarı (maxDisplacement) ve simülasyon adım sayısı (T) gibi parametreleri alıyor.

generateObjects: N tane rastgele cisim oluşturuluyor. Cisimlerin x, y, z koordinatları rastgele olarak belirleniyor.

moveObject: Belirli bir cismin rastgele yer değiştirmesini gerçekleştiriyor. Yeni koordinatlar, cismin mevcut koordinatlarından maksimum yer değiştirme miktarı içinde rastgele seçiliyor.

calculateDistance: Cisimler arasındaki mesafelerin hesaplanması için kullanılıyor.

Simulate: T adet adımda simülasyonu gerçekleştiriyor. Her adımda bir cisim seçiliyor ve moveObject fonksiyonuyla rastgele yer değiştirme yapılıyor. Cisimler arasındaki ortalama mesafe calculateAvgDistance fonksiyonuyla hesaplanıyor.

threeD: Cisimlerin 3 boyutlu uzayda görselleştirilmesini sağlıyor.

distanceGraghich: adım sayısına karşılık gelen ortalama mesafelerin grafiğini çiziyor.

Main: kullanıcıdan gerekli parametreleri alarak simülasyonu başlatıyor. Simülasyon sonucunda elde edilen verilere göre sistemin dengeye gelip gelmediği, denge durumları üzerinden cisimler arasındaki ortalama mesafe ve geçiş oranı gibi bilgileri hesaplıyor ve ekrana yazdırıyor.

Çıkarımlar:

Dengeye Gelme: Simülasyon sonucunda, sistemin dengeye gelip gelmediği kontrol edildi. Eğer sistemin dengeye geldiği belirlenirse, cisimlerin rastgele hareket etmeye devam ettiği ancak cisimler arasındaki ortalama mesafenin zaman içinde sabitlendiği görülür. Bu durumda sistemdeki cisimlerin birbirlerini etkileyerek denge durumuna ulaştığı söylenebilir.

Denge Durumlarındaki Ortalama Mesafe: Simülasyon sonucunda, denge durumlarında cisimler arasındaki ortalama mesafe hesaplandı. Dengeye gelindiğinde, cisimler arasındaki ortalama mesafenin belirli bir değerde sabitlendiği görülür. Bu değer, sistemdeki cisimlerin denge durumunda birbirlerine olan uzaklığını temsil eder.

Kod:

```
import random
import matplotlib.pyplot as plt
import math

"""
Boyutları a × b × c olan bir bölge içerisinde toplam N tane cisim vardır
(a, b, c ve N kullanıcı parametreleridir).
2.  Yukarıdaki simülasyonu yeniden değerlendirelim. 1. soruda seçilen cisme
rastgele yer değiştirme yaptırmak istediğimizde, geçmek istediği konum
yakınında dlim kadar ötede başka bir cismin olup olmadığını kontrol edelim.
Burada dlim bir parametredir. Eğer bu uzaklık içerisinde herhangi bir başka
cisim yoksa geçiş tamamlanır. Eğer bu uzaklık içerisinde başka bir cisim
varsa geçiş %50 olasılıkla tamamlanır. Bunu gerçekleştirmek için, geçilmek
istenen konumdaki en yakın diğer cismin uzaklığı hesaplanır. Örneğin dx
```

olsun ve eğer e^(-dx) ≥ R ise geçiş sağlanır. Aksi halde sağlanmaz. Burada R, [0, 1] aralığında rastgele bir sayıdır.
(a) Sistemin dengeye gelip gelmediğini kontrol ediniz.
(b) Cisimler arasındaki ortalama mesafenin zaman ile değişimine bakınız.
(c) Denge durumları üzerinden cisimler arasındaki ortalama mesafeyi bulunuz.
(d) Sistemin denge durumuna ait anlık görüntüler elde ediniz.
(e) Yeni durumlara geçiş oranını elde ediniz ve görüntüleyiniz. Geçiş oranını yaklaşık %65 civarında olması için neler yapabileceğinizi değerlendiriniz, yani çok küçük veya çok yüksek değerler olmaması için.

"""

```
# Aynı
def calculateDistance(p1, p2):
    return ((p2[0] - p1[0]) ** 2 + (p2[1] - p1[1]) ** 2 + (p2[2] - p1[2])
** 2) ** 0.5

# Aynı
def generateObjects():
    objects = []
    for i in range(N):
        x = random.uniform(0, a)
        y = random.uniform(0, b)
        z = random.uniform(0, c)
        objects.append((x, y, z))
    return objects


def moveObject(obj):
    x, y, z = obj
    new_x = random.uniform(max(x - maxDisplacement, 0), min(x +
maxDisplacement, a))
    new_y = random.uniform(max(y - maxDisplacement, 0), min(y +
maxDisplacement, b))
    new_z = random.uniform(max(z - maxDisplacement, 0), min(z +
maxDisplacement, c))

    #
    for other_obj in objects:
        if other_obj != obj:
            distance = calculateDistance((new_x, new_y, new_z), other_obj)
            if distance <= dLim:
                if random.uniform(0, 1) < 0.5:
                    dx = distance
                    prob = math.exp(-dx)
                    if prob >= R:
                        return new_x, new_y, new_z
                    else:
                        return x, y, z
    #

    return new_x, new_y, new_z

# Aynı
def calculateAvgDistance():
    total_distance = 0
    counter = 0
    for i in range(len(objects)):
        for j in range(i + 1, len(objects)):
            distance = calculateDistance(objects[i], objects[j])
            total_distance += distance
```

```
                counter += 1

        return total_distance / counter

    #
    def calculateTranRate():
        transitions = 0
        for obj in objects:
            new_x, new_y, new_z = moveObject(obj)
            if (new_x, new_y, new_z) != obj:
                transitions += 1
        transitionRate = transitions / N
        transitionRates.append(transitionRate)
    #

    def simulate():
        global objects, distances, transitionRates
        objects = generateObjects()
        distances = []
        transitionRates = []
        for i in range(T):
            randomObject = random.choice(objects)
            objects.remove(randomObject)
            objects.append(moveObject(randomObject))
            distances.append(calculateAvgDistance())
            #
            calculateTranRate()
            #

    # Aynı
    def threeD():
        fig = plt.figure()
        ax = fig.add_subplot(111, projection='3d')

        xs = [obj[0] for obj in objects]
        ys = [obj[1] for obj in objects]
        zs = [obj[2] for obj in objects]

        ax.scatter(xs, ys, zs)

        ax.set_xlabel('X')
        ax.set_ylabel('Y')
        ax.set_zlabel('Z')

        plt.show()

    # Aynı
    def distanceGraghich():
        plt.plot(range(T), distances)
        plt.xlabel('Adım Sayısı')
        plt.ylabel('Ortalama Mesafe')
        plt.show()


    def main():
        global a, b, c, N, maxDisplacement, T, dLim, R
        print("3 boyutlu evrenin büyüklük değerleri:")
        a = float(input("a: "))
        b = float(input("b: "))
        c = float(input("c: "))
        N = int(input("Kaç tane cisim olsun: "))
```

```
    maxDisplacement = float(input("Maksimum yer değiştirme miktarı: "))
    T = int(input("Simülasyon adım sayısı: "))

    #
    dLim = float(input("Geçiş kontrol uzaklığı (dlim): "))
    R = random.uniform(0, 1)
    #

    simulate()

    #
    # Sistemin dengeye gelip gelmediği
    fluctuation_threshold = 0.01
    isStable = abs(distances[-1] - distances[0]) < fluctuation_threshold
    print("Dengeye Geliş:" + str(isStable))
    #


    #
    # Denge durumları üzerinden cisimler arasındaki ortalama mesafesi
    averageDistanceInEquilibrium = sum(distances[-100:]) / 100
    print("Denge durumları üzerinden cisimler arasındaki ortalama mesafesi:
" + str(averageDistanceInEquilibrium))
    #
    threeD()
    distanceGraghich()

    # Geçiş oranının yaklaşık %65 civarında olması için dLim değeri
    desiredTransitionRate = 0.65
    RThreshold = math.log(desiredTransitionRate)
    dLim = -1 * math.log(RThreshold)  # Yeni dLim değeri
    print("Yeni dLim değeri:" + str(dLim))


main()
```

3. T büyüklüğünün etkilerini gözleyiniz.

   Yukarıdaki sistemde söz konusu hacim içerisinde cisim sayısı çok küçük veya çok büyük ise anlamlı sonuçlar elde edilemeyebilir.

   Gözlemler:

   T değeri, simülasyonun adım sayısını belirler. T'nin büyüklüğü arttıkça, sistemdeki dengelenme veya değişimlerin daha iyi anlaşılması sağlanabilir. Ancak, cisim sayısı (N) çok küçük veya çok büyük olduğunda anlamlı sonuçlar elde edilemeyebilir.

   Çok küçük cisim sayısı (N): Eğer cisim sayısı çok küçükse, sistemdeki etkileşimler ve denge durumları üzerinde istatistiksel olarak güvenilir sonuçlar elde etmek zor olabilir. Sistemin davranışı tam olarak temsil edilmeyebilir.

   Çok büyük cisim sayısı (N): Eğer cisim sayısı çok büyükse, hesaplama süresi ve bellek kullanımı artabilir. Bu da simülasyonun işlem süresini uzatabilir ve kaynakları tüketebilir. Ayrıca, çok sayıda cisim arasındaki etkileşimlerin analizi de zorlaşabilir.