# Министерство образования и науки Российской Федерации

## Федеральное государственное бюджетное образовательное учреждение высшего образования

## «Новосибирский государственный технический университет»

NSTU_Logo_blue

## Кафедра теоретической и прикладной информатики

### Лабораторная работа № 3 по дисциплине «Математические методы оптимального планирования эксперимента»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| сигма градиент синий1 | Факультет: | ПМИ |  |  |
| Группа: | ПМИМ-01 |  |  |
| Студенты: | Ершов П.К., Шибалова Ю.В. |  |  |
| Вариант: | 5 |  |  |
| Преподаватель: | Попов А.А. |  |  |

Новосибирск

2020

1. **Цель работы**

Изучить алгоритмы, используемые при построении дискретных оптимальных планов.

1. **Задание**
2. Изучить алгоритмы построения дискретных оптимальных планов.
3. Разработать программу построения дискретных оптимальных планов эксперимента, реализующую заданный алгоритм.
4. Для числа наблюдений 20, 25, 30, 35, 40 построить оптимальные планы на каждой из сеток, указанных в варианте задания. Выбрать лучшие дискретные планы для заданного числа наблюдений.
5. Оформить отчет, включающий в себя постановку задачи, результаты проведенных в п. 3 исследований, текст программы.
6. Защитить лабораторную работу.

Двухфакторная кубическая модель на квадрате со сторонами [–1, +1]. Дискретное множество – сетки 30 × 30 и 40 × 40. Строить A-оптимальные планы. Градиентный алгоритм замены. Повторные наблюдения не допускаются.

|  |  |
| --- | --- |
| Критерий | Функционал |
|  |  |

1. **Описание алгоритма работы**

Градиентный алгоритм замены не ориентирован на какой-либо определенный критерий оптимальности.

1. Выбирается невырожденный начальный план .
2. Вычисляются элементы вектора градиента

, для плана . Счетчик числа проведенных замен точек при движении по вычисленному направлению градиента устанавливается в 0 (i = 0 ).

1. Выбирается точка на множестве по правилу:

*.*

Если требуется, чтобы в плане не было повторных наблюдений, то данный поиск осуществляется на множестве .

1. Среди точек плана выбирается точка по правилу:
2. Точка заменяется в плане на точку . В результате формируется план .
3. Сравниваются величины и :
4. если > , то то счетчик i проведенных удачных замен точек увеличивается на единицу и осуществляется переход на шаг 3, при этом точки исключаются из рассмотрения;
5. в противном случае: если 0 i = , то вычисления прекращаются, иначе – s заменяется на s +1 и осуществляется переход на шаг 2.
6. **Ход работы**

**Сетка 30 x 30 Число наблюдений 20**

Функционал до оптимизации плана -129979.96066101833

Функционал после оптимизации плана -1108.0051839736273

**Сетка 30 x 30 Число наблюдений 25**

Функционал до оптимизации плана -20569.84222597912

Функционал после оптимизации плана -168.04564043555519

**Сетка 30 x 30 Число наблюдений 30**

Функционал до оптимизации плана -5995.265498711049

Функционал после оптимизации плана -184.11476508485097

**Сетка 30 x 30 Число наблюдений 35**

Функционал до оптимизации плана -1743.8455003076651

Функционал после оптимизации плана -168.63257627714074

**Сетка 30 x 30 Число наблюдений 40**

Функционал до оптимизации плана -699.5468088146511

Функционал после оптимизации плана -173.06450808313667

**Сетка 40 x 40 Число наблюдений 20**

Функционал до оптимизации плана -7099142.3642339995

Функционал после оптимизации плана -1515.1194236593242

**Сетка 40 x 40 Число наблюдений 25**

Функционал до оптимизации плана -1282917.5921654056

Функционал после оптимизации плана -3619.098597388023

**Сетка 40 x 40 Число наблюдений 30**

Функционал до оптимизации плана -349372.6859808419

Функционал после оптимизации плана -276.15797858288335

**Сетка 40 x 40 Число наблюдений 35**

Функционал до оптимизации плана -113311.22808989759

Функционал после оптимизации плана -216.052896608259

**Сетка 40 x 40 Число наблюдений 40**

Функционал до оптимизации плана -40463.93599614586

Функционал после оптимизации плана -194.65065513188807

1. **Выводы**

При увеличении числа узлов сетки на области наблюдений и числа самих наблюдений функционал так же увеличивается. Следовательно, при увеличении числа наблюдений и уменьшении шага на сетке в области наблюдений можно получить более оптимальные дискретные планы.

1. **Код программы**

**import** numpy **as** np  
  
N = 10 *# размерность информационной матрицы***def** f\_model(x, y): *# функция модели* **return** [1, x, y, x\*y, x\*\*2, y\*\*2, x\*\*2 \* y, y\*\*2 \* x, x\*\*3, y\*\*3]  
  
**def** creat\_plan(N, n): *# N - размер сетки, n - число наблюдений* step = 2 / (N - 1)  
 grid = np.arange(-1, 1 + 0.0001, step) *# сетка с шагом step* X = [] *# массив сетки* plan = [] *# массив точек плана* plan\_w = [1 / n] \* n *# массив весов плана* **for** x **in** grid: *# получаем начальный план* **for** y **in** grid:  
 X.extend([[x, y]])  
  
 **for** i **in** range(n):  
 plan.append(X[(i \* 17) % (N\*\*2)])  
 **return** X, plan, plan\_w  
  
**def** M\_mat(x\_p, p): *#Информационную матрицу M* M = np.zeros(shape = (N, N))  
 **for** i **in** range(N):  
 **for** j **in** range(N):  
 **for** k **in** range(len(x\_p)):  
 f = np.array(f\_model(x\_p[k][0], x\_p[k][1]))  
 M[i][j] += p[k] \* f[i] \* f[j]  
 **return** M  
  
**def** find\_points(M, X, flag): *# функция поиска экстремума* fi = [] *# массив значений экстремума* M1 = np.linalg.inv(M) *# получаем обратную матрицу от информационной матрицы M* M2 = np.linalg.matrix\_power(M1, 2) *# получаем обратную матрицу в квадрате (так как в формуле для поиска экстремума есть  
 # дифференциал от функцилнала критерия A по информационной матрице  
 # то, при вычислении дифференциала получаем обратную матрицу в степени 2)* **for** i **in** X: *# получаем набор чисел для поиска экстремума (в данном варианте - наибольнее число)* f\_i = np.array(f\_model(i[0], i[1])) *# получаем вектор-столбец от функции модели* buf = f\_i.T.dot(M2) *# умножаем вектор-строку от функции модели на обратную матрицу в кубе* fi.append(buf.dot(f\_i))*# умножаем получившийся в предыдущем действии вектор на вектор-столбец от функции модели* **if** flag == 0:  
 extr = max(fi)  
 **else**:  
 extr = min(fi)  
 **for** i **in** range(len(X)):  
 **if**(extr == fi[i]):  
 **return** i *# возвращаем наибольшее значение (экстремум) и точку экстремума***def** find\_point(x, X): *# проверяем принадлежность точки x массиву X* **for** i **in** X:  
 **if** x[0] == i[0] **and** x[1] == i[1]:  
 **return** 1  
 **return** 0  
  
**def** creat\_new\_multi(X, xs): *# создаём новый массив, исключая элементы массива xs* X\_new = []  
 **for** i **in** X:  
 **if** find\_point(i, xs) != 1:  
 X\_new.append(i)  
 **return** X\_new  
  
**def** funk(plan, plan\_w): *# функция получения значения функционала от криетрия оптимальности* M = M\_mat(plan, plan\_w)  
 M1 = np.linalg.inv(M) *# получаем обратную матрицу от информационной матрицы M* F = -np.trace(M1)  
 **return** (F)  
  
  
**def** building(plan, plan\_w, X): *# функция построения оптимального плана* i = 0 *# счётчик успешных замен точек* X\_res = X.copy() *# копируем массив сетки* new\_plan = plan.copy() *# копируем точки плана* new\_plan\_i = new\_plan.copy() *# создаём следующую копию точек плана для полученния нового значения функционала* **while**(**True**):  
 M = M\_mat(new\_plan\_i, plan\_w)  
 old\_funk = funk(new\_plan\_i, plan\_w) *# получаем старое значение функционала* **if**(i == 0): *# если ноль замен, то обновляем план* new\_plan = new\_plan\_i.copy()  
  
 X\_new = creat\_new\_multi(X\_res, new\_plan\_i) *# получаем новую сетку, без точек плана* X\_max = find\_points(M, X\_new, 0) *# максимум на множестве X \ plan (x\*)* X\_min = find\_points(M, new\_plan, 1) *# минимум на множестве plan (x\*\*)* old\_plan = new\_plan\_i.copy() *# сохраняем старый план  
 # заменям точку x\*\* на точку x\** x\_star\_2 = new\_plan\_i[X\_min]  
 new\_plan\_i[X\_min] = X\_res[X\_max]  
  
 new\_funk = funk(new\_plan\_i, plan\_w) *# получаем новый функционал* **if** (new\_funk > old\_funk): *# пункт 6 алгоритма* i += 1 *# увеличиваем счётких замен точек* del\_point = X\_res[X\_max] *# исключаем точку x\* из рассмотрения* X\_res.remove(del\_point)  
 **if** (x\_star\_2 **in** X\_res): *# исключаем точку x\*\* из рассмотрения* X\_res.remove(x\_star\_2)  
 **else**:  
 **if** (i == 0): *# больше ничего нельзя сделать, завершить вычисления (подпункт b пункта 6)* **return** new\_plan  
 **else**: *# возможно еще можно найти "удачные" замены (переход к пункту 2)* new\_plan\_i = old\_plan.copy()  
 i = 0  
  
**def** main():  
 g = [30, 40]  
 obser = [20, 25, 30, 35, 40]  
  
 **for** i **in** g:  
 **for** j **in** obser:  
 print(**"Сетка"**, i, **"x"**, i, **"Число наблюдений"**, j)  
 X, plan, plan\_w = creat\_plan(i, j)  
 print(**"Функционал до оптимизации плана"**, funk(plan, plan\_w))  
 optimal\_plan = building(plan, plan\_w, X)  
 print(**"Функционал после оптимизации плана"**, funk(optimal\_plan, plan\_w), **"\n"**)  
  
main()