МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное

учреждение высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»



Факультет прикладной математики и информатики

Кафедра ТПИ

Дисциплина: «Методы активной идентификации динамических систем»

Лабораторная работа №1

Уровень 2, вариант №2

**ОЦЕНИВАНИЕ НЕИЗВЕСТНЫХ ПАРАМЕТРОВ**

Факультет: ФПМИ

Группа: ПМИМ-31

Выполнили: Монгуш Н. С., Тарулин М. А., Филипенко Ю. Д.

Преподаватель: Чубич В. М.

Дата выполнения:

Отметка о защите:

1. **Цель работы**

Научиться применять метод максимального правдоподобия при оценивании неизвестных параметров моделей линейных дискретных систем.

1. **Задание**
2. Разработать программу вычисления критерия идентификации и его градиента.
3. Разработать программу нахождения оценок максимального правдоподобия.

3. Следуя своему варианту задания, для указанных истинных значений параметров компьютерным моделированием получить последовательность из 30 измерений, соответствующую указанному входному сигналу. Используя полученные данные наблюдений, вычислить оценки максимального правдоподобия. Для ослабления зависимости результатов оценивания от выборочных данных, осуществить и обработать пять подобных идентификационных экспериментов, запоминая полученные результаты. Усреднив , найти .

**3. Постановка задачи**

Исследуемая модель имеет вид:

где – n-вектор состояния;

– детерминированный r-вектор управления (входа);

– m-вектор измерения (выхода);

– m-вектор шума (ошибки) измерения.

Вычисление оценки максимального правдоподобия является задачей оптимизации следующего вида:

где – логарифмическая функция правдоподобия,

– критерий идентификации, который определяется следующим образом:

где – число испытаний;

– количество независимых запусков системы;

*–* m-мерный вектор обновления в момент времени:

Алгоритм вычисления значения критерия идентификации при некотором фиксированном :

1. Определить .
2. Положить .
3. Положить
4. Определить .
5. Выбрать если .
6. Вычислить .
7. Положить .
8. Вычислить при помощи соотношения (5).
9. Положить .
10. Увеличить на единицу. Если , перейти на шаг 8.
11. Увеличить на единицу. Если , перейти на шаг 4.
12. Положить .
13. Увеличить на единицу. Если , перейти на шаг 3.
14. Положить и закончить процесс.

Алгоритм вычисления градиентов критерия максимального правдоподобия при фиксированном :

1. Определить и .
2. Положить .
3. Положить .
4. Определить .
5. Выбрать если .
6. Вычислить .
7. Найти .
8. Положить .
9. Вычислить при помощи соотношения (5).
10. Положить .
11. Увеличить на единицу. Если , перейти на шаг 9.
12. Увеличить на единицу. Если , перейти на шаг 4.
13. Положить .
14. Увеличить на единицу. Если , перейти на шаг 3, иначе закончить процесс.

Относительная ошибка оценивания в пространстве параметров вычисляется по формуле:

где - истинные значения параметров;

– средняя оценка значений параметров по n наблюдениям.

Для оценки относительной ошибки оценивания в пространстве откликов используется соотношение:

где

.

При этом находится при θ = в соответствии с равенством:

**4. Исходные данные**

|  |  |  |  |
| --- | --- | --- | --- |
| Матрицы для моделей состояния и измерения | Ковариационные матрицы шумов и начальные условия |  |  |
|  |  |  |  |

**5. Полученный результат**

В ходе каждого эксперимента было смоделировано 5 последовательностей из 30 наблюдений. Для каждой последовательности были получены оценки неизвестных параметров модели метода нулевого и первого порядка.

Таблица 1. Результаты эксперимента.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Номер эксперимента | Нулевой порядок | | Первый порядок | |
|  |  |  |  |
| 1 | -2 | -0.05 | -1.291 | -0.049 |
| 2 | -0.437 | -0.05 | -0.282 | -0.049 |
| 3 | -1.888 | -0.05 | -1.218 | -0.045 |
| 4 | -0.767 | -0.05 | -0.495 | -0.0458 |
| 5 | -1.100 | -0.05 | -0.710 | -0.0500 |
|  | -1.100 | -0.05 | -0.710 | -0.05 |
|  | 0.623 | | 0.729 | |
|  | 0.987 | | 1.000 | |
|  |  | |  | |

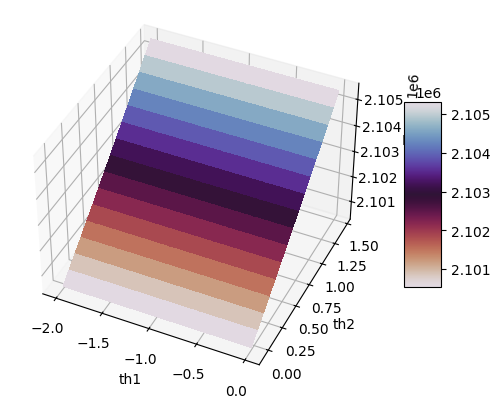


Рисунок 1. Значение критерия идентификации в зависимости от неизвестных параметров.

**6. Код программы**

**import** **numpy** **as** **np**

**from** **numpy.linalg** **import** norm

**from** **numpy.random** **import** normal

**from** **scipy.optimize** **import** minimize, Bounds, least\_squares

**import** **matplotlib.pyplot** **as** **plt**

**from** **matplotlib** **import** cm

**def** **F\_**(th\_1: float) -> np.matrix:

**return** np.matrix([[-**0.8**, **1**], [th\_1, **0**]])

**def** **Psi\_**(th\_2: float) -> np.matrix:

**return** np.matrix([[th\_2], [**1**]])

G = np.matrix([**1**, **1**]).T

H: np.matrix = np.matrix([[**1**, **0**]])

R: np.ndarray = np.array([[**0.1**]])

Q = np.matrix(**0.1**)

x\_0: np.matrix = np.matrix([[**0**], [**0**]])

th\_true: np.ndarray = np.array([-**1.5**, **1**])

th\_1\_range: list = [-**2**, -**0.05**]

th\_2\_range: list = [**0.01**, **1.5**]

u\_tk: np.matrix = np.matrix([[**1**]])

N: int = **30** # Размерность сигнала U

s: int = **2** # Размерность th

U: np.matrix = np.matrix(np.ones(N+**1**)\***2**).T

**def** **gen\_XY**(th: np.ndarray, err=**1**, err2=**1**):

X: np.matrix = np.matrix(np.zeros((**2**, N+**1**)))

Y: np.matrix = np.matrix(np.zeros((N+**1**, **1**)))

F: np.matrix = F\_(th[**0**])

Psi: np.matrix = Psi\_(th[**1**])

X[:, **0**] = x\_0

Y[**0**] = H @ X[:, **0**] + normal(-np.sqrt(R)/**2**, np.sqrt(R))\*err

**for** k **in** range(**1**, N+**1**):

wi = normal(-np.sqrt(R)/**2**, np.sqrt(Q), (**1**,**2**))\*err

vi = normal(-np.sqrt(Q)/**2**, np.sqrt(R))\*err\*err2

X[:, k] = F @ X[:, k-**1**] + Psi @ U[k] + (G @ wi)[**0**].T

Y[k] = H @ X[:, k] + vi

**return** X, Y

X, Y = gen\_XY(th\_true)

**def** **Hi**(th: np.ndarray, Y: np.ndarray) -> float:

m: int = **1**

v: int = **1**

F: np.matrix = F\_(th[**0**])

Psi: np.matrix = Psi\_(th[**1**])

Hi: float = N \* m \* np.log(**2** \* np.pi) + N \* v \* np.log(np.linalg.det(R))

x\_k: np.ndarray = x\_0

**for** k **in** range(N):

x\_k\_k: np.ndarray = F @ x\_k + Psi @ u\_tk

epsilon: np.ndarray = Y[k + **1**] - H @ x\_k\_k

delta = epsilon.T @ np.linalg.inv(R) @ epsilon

Hi += delta

**return** **0.5** \* Hi.item()

Hi(th\_true, Y)

**def** **gradHi**(th: np.ndarray, Y: np.ndarray) -> float:

dF: list[np.matrix, np.matrix] = [np.matrix([[**0**, **0**], [**0**, **0**]]), np.matrix([[**0**, **0**], [**1**, **0**]])]

dPsi: list[np.matrix, np.matrix] = [np.matrix([[**0**], [**0**]]), np.matrix([[**1**], [**0**]])]

dx\_0: list[np.matrix, np.matrix] = [np.matrix([[**0**], [**0**]]), np.matrix([[**0**], [**0**]])]

v: int = **1**

F: np.matrix = F\_(th[**0**])

Psi: np.matrix = Psi\_(th[**1**])

gradHi: np.ndarray = np.zeros(len(th))

x\_k: np.ndarray = x\_0

dx\_k: list[np.matrix, np.matrix] = dx\_0

delta = np.zeros((s, **1**))

R\_inv = np.linalg.inv(R)

**for** k **in** range(N):

x\_k\_k: np.ndarray = F @ x\_k + Psi @ u\_tk

dx\_k\_k: list = [None, None]

dEpsilon: list = [None, None]

**for** i **in** range(s):

dx\_k\_k[i]: np.matrix = dF[i] @ x\_k + F @ dx\_k[i] + dPsi[i] @ u\_tk

dEpsilon[i]: np.matrix = - H @ dx\_k\_k[i]

epsilon: np.ndarray = Y[k + **1**] - H @ x\_k\_k

delta[i] += (dEpsilon[i].T @ np.linalg.inv(R) @ epsilon).item()

gradHi[i] += delta[i].item()

**return** gradHi

th\_est\_m0 = []

th\_est\_m1 = []

Y\_observation\_list = []

**for** i **in** range(**5**):

\_, Y\_obs = gen\_XY(th\_true)

Y\_observation\_list.append(Y\_obs)

th\_0 = np.array([np.random.uniform(-**2**, -**0.05**), np.random.uniform(**0.01**, **1.5**)])

res\_0 = minimize(

method="nelder-mead",

fun=Hi,

x0=th\_0,

bounds=Bounds([-**2**, -**0.05**], [**0.01**, **1.5**]),

options={"xatol": **1e-5**, "fatol": **1e-5**},

tol=**1e-5**,

args=(Y\_obs, ),

)

res\_1 = minimize(

method="SLSQP",

fun=Hi,

x0=th\_0,

jac=gradHi,

bounds=Bounds([-**2**, -**0.05**], [**0.01**, **1.5**]),

options={"eps": **1e-12**, "ftol": **1e-12**},

tol=**1e-12**,

args=(Y\_obs, ),

)

**print**(res\_0.x, res\_1.x)

**print**(f"th0: {th\_0}**\t**М0: {res\_0['x']} ({res\_0['nit']}/{res\_0['nfev']})**\**

М1: {res\_1['x']} ({res\_1['nfev']}/{res\_1['njev']}/{res\_1['nit']})")

th\_est\_m0.append(res\_0['x'])

th\_est\_m1.append(res\_1['x'])

th\_est\_m0 = np.array(th\_est\_m0)

th\_est\_m1 =np.array(th\_est\_m1)

th\_est\_m0 = np.mean(th\_est\_m0[:, **0**]), np.mean(th\_est\_m0[:, **1**])

th\_est\_m1 = np.mean(th\_est\_m1[:, **0**]), np.mean(th\_est\_m1[:, **1**])

**print**(f"Среднее по 5 наблюдениям**\n**M0: {th\_est\_m0}**\n**M1: {th\_est\_m1}")

**def** **mean**(th):

Y\_ = np.array([gen\_XY(th, **1**, **0**)[**1**] **for** \_ **in** range(**100**)])

**return** np.array([np.mean(np.array(Y\_)[:, i]) **for** i **in** range(N+**1**)]).reshape(N+**1**)

Y\_mean\_obs = np.array([np.mean(np.array(Y\_observation\_list)[:, i]) **for** i **in** range(N+**1**)])

Y\_est\_m0 = mean(th\_est\_m0)

Y\_est\_m1 = mean(th\_est\_m1)

dth\_m0 = norm(th\_true - th\_est\_m0) / norm(th\_true)

dY\_m0 = norm(Y\_mean\_obs - Y\_est\_m0) / norm(Y\_mean\_obs)

dth\_m1 = norm(th\_true - th\_est\_m1) / norm(th\_true)

dY\_m1 = norm(Y\_mean\_obs - Y\_est\_m1) / norm(Y\_mean\_obs)

**print**("0-го порядка:")

**print**(f"Ошибка в пространстве параметров: {dth\_m0}")

**print**(f"Ошибка в пространстве отклика: {dY\_m0}")

**print**("1го порядка:")

**print**(f"Ошибка в пространстве параметров: {dth\_m1}")

**print**(f"Ошибка в пространстве отклика: {dY\_m1}")

Y\_true = gen\_XY(th\_true, **0**, **0**)[**1**]

step = **0.1**

th1a = np.arange(-**2**, **0.01**, step)

th2a = np.arange(-**0.05**, **1.499**, step)

th1, th2 = np.meshgrid(th1a, th2a)

ci = np.array([[Hi(np.array([th1\_, th2\_]), Y\_true) **for** th1\_, th2\_ **in** zip(a,b)] **for** a,b **in** zip(th1, th2)])

fig, ax = plt.subplots(subplot\_kw={"projection": "3d"})

surf = ax.plot\_surface(th1, th2, ci, cmap=cm.twilight\_r, linewidth=**0**, antialiased=False)

ax.view\_init(**40**, **295**);

ax.set\_xlabel('th1')

ax.set\_ylabel('th2')

ax.set\_zlabel('HI')

fig.colorbar(surf, shrink=**0.5**, aspect=**5**);

fig.show()