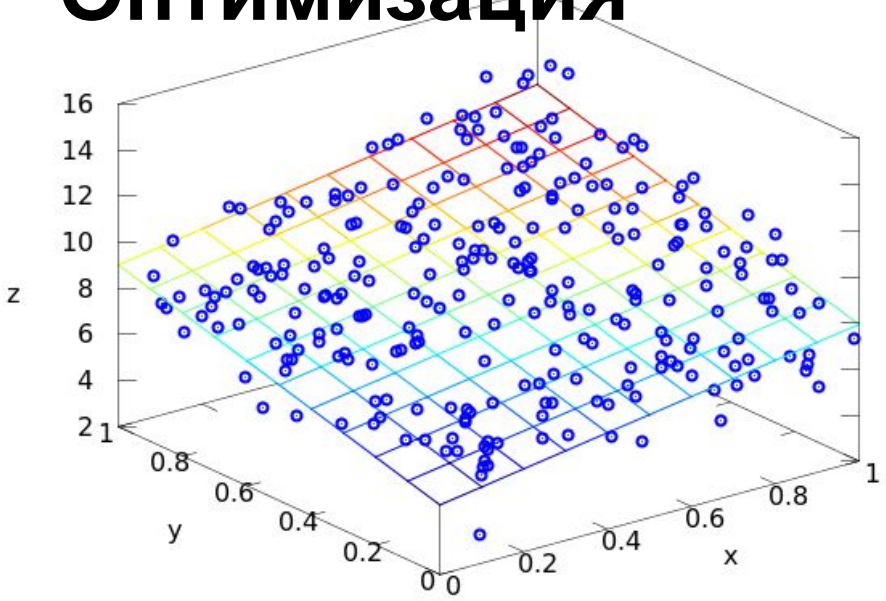
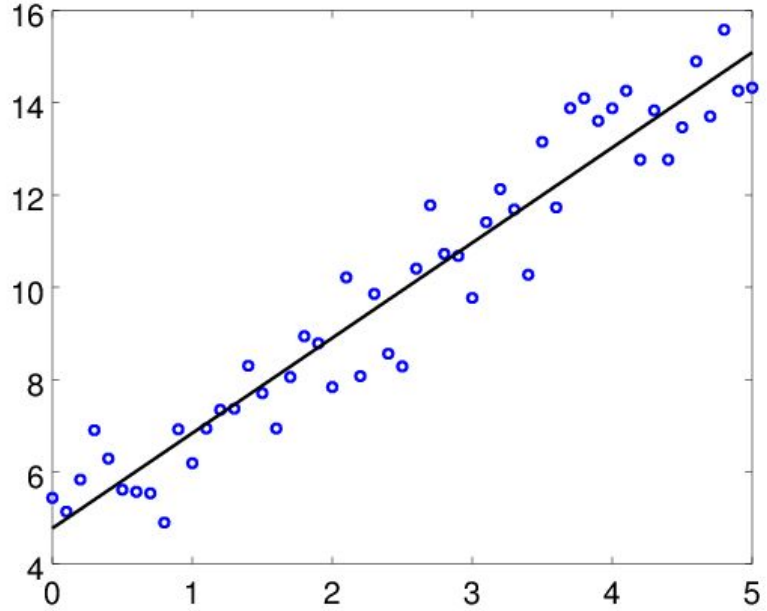
Занятие 5. Оптимизация



Краткое содержание

1. Линейная регрессия
2. Нелинейная регрессия и функция `lsqnonlin`
3. Системы нелинейных уравнений и функция `fsolve`

Часть 1. Линейная регрессия (метод наименьших квадратов)



Линейная регрессия

Исходные данные
Точки в (k+1) – мерном пространстве
$(y_i, x_{i1}, \dots, x_{ik})$

+

Аппроксимирующая функция
$y = \beta_0 + \beta_1 x_{i1} + \cdots + \beta_k x_{ik}$
β – параметры модели

Система уравнений (переопределённая):

$$\begin{cases} \beta_1 x_{11} + \beta_2 x_{12} + \cdots + \beta_k x_{1k} = y_1 \\ \beta_1 x_{21} + \beta_2 x_{22} + \cdots + \beta_k x_{2k} = y_2 \\ \dots \\ \beta_1 x_{n1} + \beta_2 x_{n2} + \cdots + \beta_k x_{nk} = y_n \end{cases}$$

Система в матричном виде:

$$\boldsymbol{X\beta = y}$$

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nk} \end{bmatrix}; y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix};$$

$$\beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix}$$

Линейная регрессия: метод наименьших

Система в матричном виде:

$$\boldsymbol{X\beta = y}$$

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1k} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nk} \end{bmatrix}; y = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}; \beta = \begin{pmatrix} \beta_1 \\ \vdots \\ \beta_n \end{pmatrix}$$

Сумма квадратов отклонений

$$RSS = \sum_i (y_i - \hat{y}_i)^2 = \sum_i e_i^2$$
$$= e^\top e = (y - X\beta)^\top (y - X\beta) =$$
$$= y^\top y - 2y^\top X\beta + \beta^\top X^\top X\beta$$

Поиск минимума

$$\frac{\partial RSS}{\partial \beta} = -2\frac{\partial (y^\top X\beta)}{\partial \beta} + \frac{\partial (\beta^\top X^\top X\beta)}{\partial \beta} = 0$$

$$\frac{\partial RSS}{\partial \beta} = -2X^\top y + 2X^\top X\beta = 0$$

$$X^\top X\beta = X^\top y \Leftrightarrow \hat{\beta} = (X^\top X)^{-1} X^\top y$$

Несмещённая оценка ошибки регрессии

$$\hat{\sigma}^2 = \frac{1}{n-m}\sum_i e_i^2 = \frac{e^\top e}{n-m}$$

$$e = y - \hat{y}$$

n – число точек, m – число коэффициентов регрессии

Ковариационная матрица

Несмещённость оценок параметров регрессии

$$E[\hat{\beta}] = E[(X^{\mathsf{T}}X)^{-1}X^{\mathsf{T}}(XB + \varepsilon)] = E[B] + (X^{\mathsf{T}}X)^{-1}X^{\mathsf{T}}E[\varepsilon] = E[B]$$

B – истинное значение параметров регрессии, ε – вектор ошибок

Ковариационная матрица

$$\operatorname{cov}(\hat{\beta},\hat{\beta}) = E\left[(\hat{\beta} - B)(\hat{\beta} - B)^{\mathsf{T}}\right] = E[(X^{\mathsf{T}}X)^{-1}X^{\mathsf{T}}\varepsilon\varepsilon^{\mathsf{T}}X(X^{\mathsf{T}}X)^{-1}] =$$
$$= (X^{\mathsf{T}}X)^{-1}X^{\mathsf{T}}E[\varepsilon\varepsilon^{\mathsf{T}}]X(X^{\mathsf{T}}X)^{-1} = \hat{\sigma}^2(X^{\mathsf{T}}X)^{-1}$$

B – истинное значение параметров регрессии, ε – вектор ошибок

Вид ковариационной матрицы

$$\operatorname{cov}(\hat{\beta},\hat{\beta}) = \begin{pmatrix} \operatorname{Var}[\hat{\beta}_1] & \operatorname{cov}(\hat{\beta}_1,\hat{\beta}_2) & \cdots & \operatorname{cov}(\hat{\beta}_1,\hat{\beta}_m) \\ \operatorname{cov}(\hat{\beta}_1,\hat{\beta}_2) & \operatorname{Var}[\hat{\beta}_2] & \cdots & \operatorname{cov}(\hat{\beta}_2,\hat{\beta}_m) \\ \vdots & \vdots & \ddots & \vdots \\ \operatorname{cov}(\hat{\beta}_m,\hat{\beta}_1) & \operatorname{cov}(\hat{\beta}_m,\hat{\beta}_2) & \cdots & \operatorname{Var}[\hat{\beta}_m] \end{pmatrix}$$

Доверительные интервалы

$$s^2_{\beta_i} = \operatorname{Var}[\widehat{\beta_i}]$$

$$\Delta\hat{\beta}_i = s_{\beta_i} \cdot t(\alpha, f)$$

t – двухсторонний квантиль t-распределения;

α – вероятность, $f = n - m$ – число степеней свободы

Доверительный интервал и интервал предсказания

Доверительный интервал $\hat{y}$ (confidence interval)

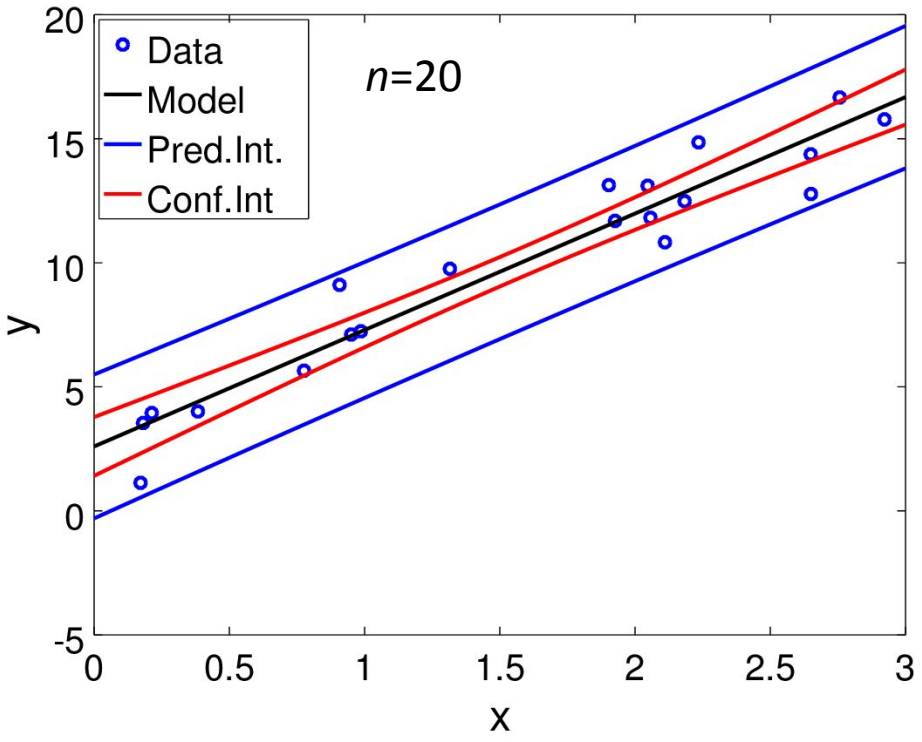
$$\hat{y} \pm \hat{\sigma} t_{\alpha,n-m} \sqrt{x^\top (X^\top X)^{-1} x}$$

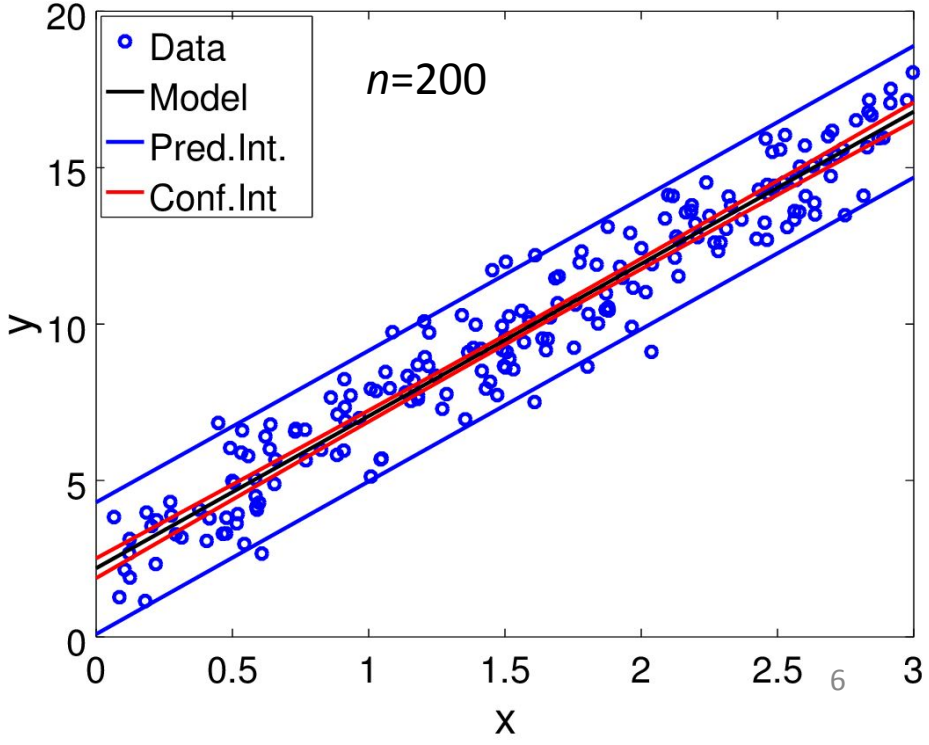
Исходная функция с вероятностью 95% проходит через этот интервал

Интервал предсказания (prediction interval)

$$\hat{y} \pm \hat{\sigma} t_{\alpha,n-m} \sqrt{1 + x^\top (X^\top X)^{-1} x}$$

Новая точка попадёт в этот интервал с вероятностью 95%





Задача: нахождение коэффициентов регрессии

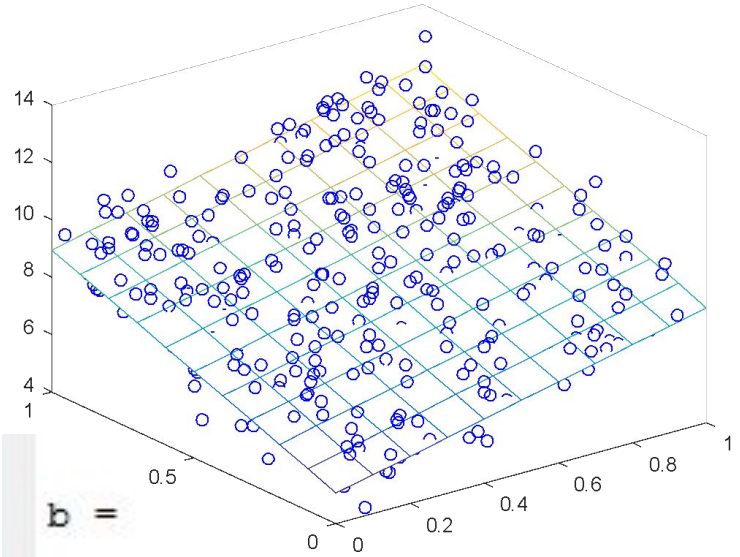
$$\begin{cases} ax_1 + by_1 + c = z_1 \\ \dots \\ ax_1 + by_1 + c = z_1 \end{cases} \Leftrightarrow \begin{pmatrix} x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots \\ x_n & y_n & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} y_1 \\ \dots \\ y_n \end{pmatrix} \Leftrightarrow X\beta = y$$

Т.к. матрица X – не квадратная, то записать $\beta = X^{-1}y$ нельзя

Но MATLAB решит эту систему уравнений, если написать `b=X\y`

Решение

```
% Создание выборки точек
x = rand(500, 1);
y = rand(500, 1);
z = 3*x+4*y+5+randn(size(x));
plot3(x,y,z,'bo');
% Решение системы уравнений
X = [x y ones(size(x))];
b = X \ z
[Xm,Ym]=meshgrid(0:0.1:1);
Zm = b(1)*Xm + b(2)*Ym + b(3);
hold on;
mesh(Xm,Ym,Zm);
hold off;
```




```
b =

    2.9432
    3.8901
    5.0551
```

Задача: доверительные интервалы значений $\hat{\beta}$

Шаг 1. Ошибка регрессии

```
>> res = z-(b(1)*x+b(2)*y+b(3));
>> f = numel(res) - numel(b);
>> sigma2 = res'*res/f
   sigma2 =  0.808416630656864
```

Шаг 2. Ковариационная матрица

```
>> C = sigma2 * inv(X'*X)
C =
  0.0204893 -0.0013650 -0.0096530
 -0.0013650  0.0188808 -0.0085331
 -0.0096530 -0.0085331  0.0106459
```

Внимание! $r(\hat{\beta}_i, \hat{\beta}_j) \neq 0$

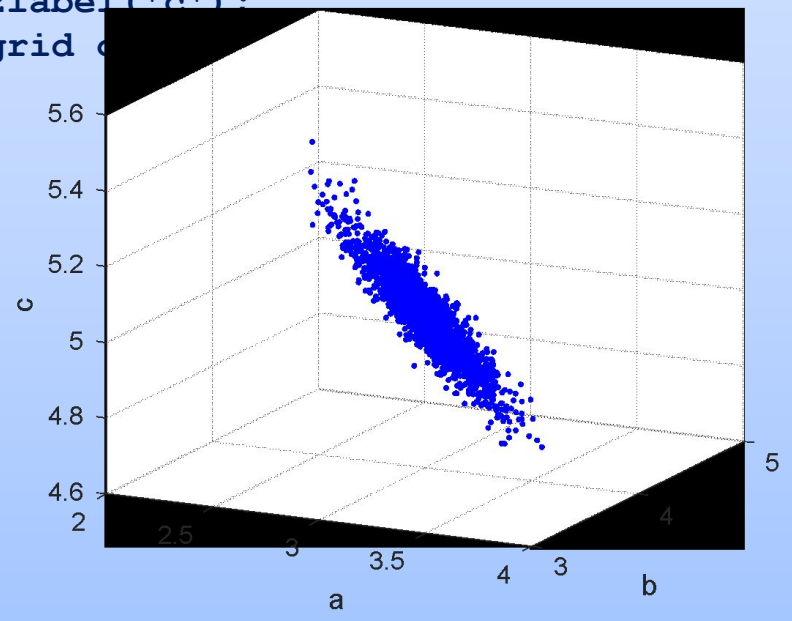
Оставляйте «запасные» знаки при округлении $\hat{\beta}$!

Шаг 3. Ошибки коэффициентов

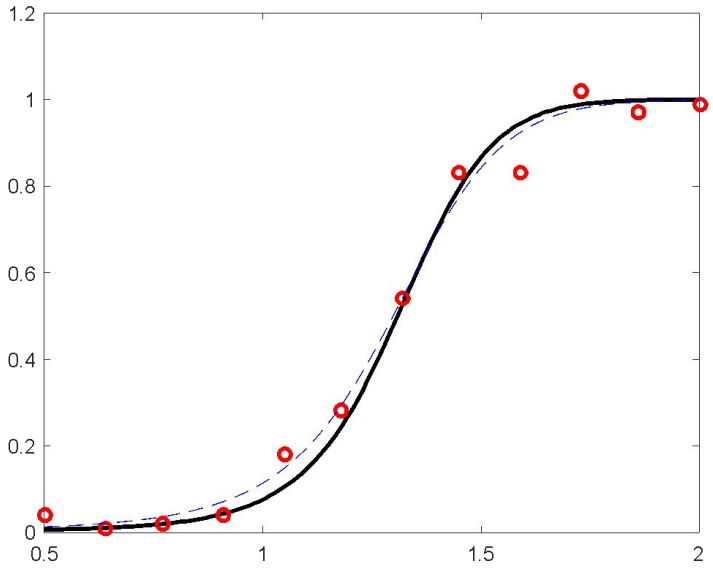
```
>> sb = sqrt(diag(C))
    0.14314    0.13741    0.10318
>> db = sb * tinv(1-0.05/2,f)
    0.28124    0.26997    0.20272
```

Параметры коррелированы!

```
bm = nan(3,2000);
for i = 1:2000
  x = rand(500, 1);
  y = rand(500, 1);
  z = 3*x+4*y+5+randn(size(x));
  bm(:,i)=[x y ones(size(x))]\z;
end
plot3(bm(1,:),bm(2,:),bm(3,:),'b.')
xlabel('a'); ylabel('b');
zlabel('c');
grid
```



Часть 2. Нелинейная регрессия (метод наименьших квадратов)



Нелинейный метод наименьших

1. Нелинейная система уравнений (переопределённая)

$$\begin{cases} \varphi(\beta_1, \dots, \beta_k, x_{11}, x_{21}, \dots x_{m1}) = y_1 \\ \varphi(\beta_1, \dots, \beta_k, x_{12}, x_{22}, \dots, x_{m2}) = y_2 \\ \dots \\ \varphi(\beta_1, \dots, \beta_k, x_{1n}, x_{2n}, \dots, x_{mn}) = y_n \end{cases}$$

2. Минимизируемая сумма квадратов

$$F(\vec{\beta}) = \sum_i (\hat{y}_i - y_i)^2$$

$$= \sum_i \left(\varphi(\vec{\beta}, \vec{x}_i) - y_i\right)^2 = \sum_i f_i^2$$

3. Система уравнений для поиска минимума суммы

$$\begin{cases} \dfrac{\partial F(\vec{\beta})}{\partial \beta_1} = 0 \\ \dots \\ \dfrac{\partial F(\vec{\beta})}{\partial \beta_m} = 0 \end{cases}$$

Как правило, полученную систему решается только численными методами (не аналитическими)

1. Методы Ньютона и Гаусса-Ньютона
2. Метод Левенберга-Марквардта
3. Методы доверительных областей (trust region)

МНК и метод Гаусса-Ньютона

Система уравнений и метод Ньютона

$$\begin{cases} \dfrac{\partial F(\vec{\beta})}{\partial \beta_1} = 0 \\ \cdots \\ \dfrac{\partial F(\vec{\beta})}{\partial \beta_m} = 0 \end{cases} \Rightarrow \frac{\partial F(\vec{\beta})}{\partial \beta_i} \approx \frac{\partial F(\vec{\beta}^{(0)})}{\partial \beta_i} + \sum_j \frac{\partial F(\vec{\beta}^{(0)})}{\partial \beta_i \partial \beta_j}\left(\beta_j - \beta_j^{(0)}\right) = 0$$

Матричная запись и метод Гаусса-Ньютона

$$\nabla F(\vec{\beta}^{(0)}) + H(\vec{\beta}^{(0)})\vec{p} = 0 \Rightarrow J^{\top}(\vec{\beta}^{(0)})f(\vec{\beta}^{(0)}) + J^{\top}(\vec{\beta}^{(0)})J(\vec{\beta}^{(0)})\vec{p} = 0$$

Градиент, якобиан и гессиан

$$(\nabla F)_i = \frac{\partial F}{\partial \beta_i} = \frac{\partial}{\partial \beta_i}\left(\sum_j f_j^2\right) = 2\sum_j f_j \frac{\partial f_j}{\partial \beta_i} = 2J^{\top}f; J_{ij} = \frac{\partial f_i}{\partial \beta_j} = \frac{\partial \varphi(\vec{\beta}, \vec{x_i})}{\partial \beta_j}$$

$$H_{ij} = \frac{\partial F(\beta)}{\partial \beta_i \partial \beta_j} = \frac{\partial}{\partial \beta_j}\left[2\sum_k f_k \frac{\partial f_k}{\partial \beta_i}\right] = 2\sum_k \left[\frac{\partial f_k}{\partial \beta_i}\frac{\partial f_k}{\partial \beta_j} + f_k \frac{\partial f_k}{\partial \beta_i \partial \beta_j}\right] \approx 2\sum_k \frac{\partial f_k}{\partial \beta_i}\frac{\partial f_k}{\partial \beta_j} = 2J^{\top}J$$

Метод Левенберга-

$$(J^\top J + \lambda I)\vec{p} = -\nabla F = -J^\top f \text{ (вариант 1)}$$

$$(J^\top J + \lambda \mathrm{diag}[J^\top J])\,\vec{p} = -\nabla F = -J^\top f \text{ (вариант 2)}$$

Каждая итерация включает в себя подбор шага λ:

1. Взять начальное (очень малое) значение λ
2. Выполнить итерацию, найдя $\vec{p}$
3. Если значение F возросло, то увеличить шаг λ и вернуться к п.1
4. Если значение F уменьшилось, то принять полученное значение $\vec{p}$ и перейти к следующей итерации

Сходится медленнее метода Гаусса-Ньютона, но менее требователен к начальному приближению. Широко применяется на практике.

Метод Л.-М. – комбинация методов Гаусса-Ньютона и градиентного спуска

а) метод градиентного спуска: $\vec{p} = -\lambda \nabla F\left(\vec{\beta}^{(0)}\right)$

б) метод Гаусса-Ньютона: $J^\top J\vec{p} = -\nabla F$

Обозначения: $\vec{p} = \vec{\beta} - \vec{\beta}^{(0)}$, $\vec{\beta}^{(0)}$ – начальное приближение; $\vec{\beta}$ – результат итерации, λ – шаг, F – минимизируемая функция

Нелинейная регрессия: доверительные интервалы

Линейная регрессия

$$s^2_{\hat{\boldsymbol{\beta}}} = \hat{\sigma}^2 (X^\top X)^{-1}$$

$$X = \begin{pmatrix} 1 & x_1^{(1)} & \cdots & x_m^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(n)} & \cdots & x_m^{(n)} \end{pmatrix}$$

Нелинейная регрессия

$$s^2_{\hat{\boldsymbol{\beta}}} = \hat{\sigma}^2 (J^\top J)^{-1}$$

При этом якобиан и отклонения рассчитываются в точке $\hat{\boldsymbol{\beta}}$

Исходная система уравнений

$$\begin{cases} \varphi(\beta, x_1) = y_1 \\ \cdots \\ \varphi(\beta, x_n) = y_n \end{cases}$$

Результат линеаризации в векторной форме

$$J(\hat{\beta})(\beta - \hat{\beta}) = y - \hat{y} = e$$

Разложение в ряд Тейлора

$$\varphi(\beta, x) = \varphi(\hat{\beta}, x) + \sum_i \frac{\partial \varphi(\hat{\beta}, x)}{\partial \hat{\beta}_i} (\beta_i - \widehat{\beta_i})$$

$J(\hat{\beta})$ - аналог матрицы X в линейной регрессии

Нелинейная регрессия: практическая реализация

Шаг 1. Задание аппроксимирующей функции

$$y = \beta_1 + \beta_2 \exp(-\beta_3 x)$$

Шаг 2. Задание якобиана

$$\begin{cases} \dfrac{\partial y_i}{\partial \beta_1} = 1 \\ \dfrac{\partial y_i}{\partial \beta_2} = \exp(-\beta_3 x_i) \\ \dfrac{\partial y_i}{\partial \beta_3} = -x\beta_2 \exp(-\beta_3 x_i) \end{cases}$$

Шаг 3. Запись на MATLAB

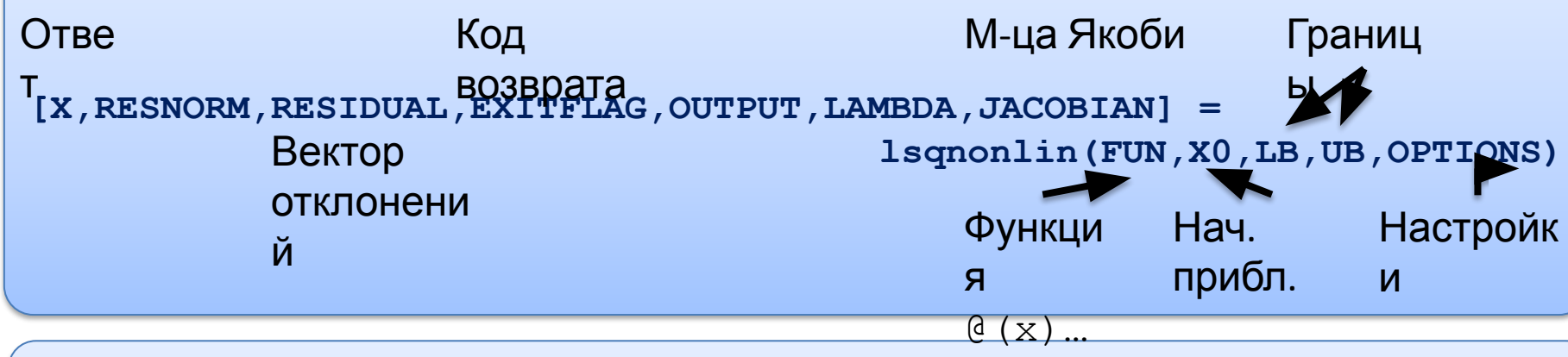
```
function lsqfit_ex
[...данные...]
b0 = [0 3 2];
opt = optimset('Display','iter',...
  'Jacobian', 'on', ...
  'DerivativeCheck', 'on');
[b,~,res,~,~,~,J] = lsqnonlin(@(b)
func(b, X, Y), b0, [], [], opt);
[...анализ и вывод результатов...]
end

function [dF, J] = func(b, x, y)
dF = b(1) + b(2)*exp(-b(3)*x) - y;
df_db1 = ones(size(x));
df_db2 = exp(-b(3)*x);
df_db3 = -b(2)*exp(-b(3)*x).*x;
J = [df_db1 df_db2 df_db3];
end
```

Шаг 4. Подбор начального приближение и визуализация результатов

Функции lsqnonlin и optimset:

Функция lsqnonlin – реализация метода наименьших квадратов



Функция optimset – настройки для lsqnonlin

```
OPT = optimset('param1', value1, 'param2', value2, ...);
```

Параметр	Описание
Algorithm	Используемый алгоритм
DerivativeCheck	Проверка якобиана (‘on’/’off’)
Display	Отладочная печать (‘iter’/’final’/’off’ и др.)
Jacobian	Использовать пользовательский якобиан (‘on’/off’)
TolFun	Минимальное изменение функции
TolX	Минимальное изменение параметров

Часть 3. Системы уравнений

```
>> fsolve_ex
```

```
                                                                    
   DerivativeCheck Information

Objective function derivatives:
Maximum relative difference between user-supplied
and finite-difference derivatives = 1.47737e-08.

DerivativeCheck successfully passed.
                                                                    
```

Iteration	Func-count	f(x)	Norm of step	First-order optimality	Trust-region radius
0	1	0.919395		1.23	1
1	2	0.550436	0.775422	3.09	1
2	3	0.0196684	0.164206	0.387	1
3	4	7.818e-05	0.0504667	0.0206	1
4	5	1.05649e-09	0.00339965	7.61e-05	1
5	6	2.06816e-19	1.27224e-05	1.06e-09	1

Equation solved.

Решение систем нелинейных уравнений

Обычно для решения системы линейных уравнений используются те же алгоритмы, что и для метода наименьших квадратов. В MATLAB – методы Левенберга-Марквардта и доверительных областей (trust-region dogleg)

Функция `fsolve` – численное решение системы уравнений

```
[X,FVAL,EXITFLAG,OUTPUT,JACOB] = fsolve(FUN,X0,OPTIONS)
```

Ответ — Вектор отклонений

Ф-ция `@(x)…` — Нач. прибл. — Настройки

Пример

$$\begin{cases} x^3 + \cos y - 2 = 0 \\ \sin x^2 + \ln y = 0 \end{cases}$$

$$J = \left(\frac{\partial F_i}{\partial \beta_j}\right) = \begin{pmatrix} 3x^2 & -\sin y \\ 2x\cos x^2 & y^{-1} \end{pmatrix}$$

Решение без якобиана

```
>> f=@(p)[p(1)^3+cos(p(2))-2;
   sin(p(1)^2) + log(p(2))];
>> [p,ff] = fsolve(f,[1 1])
Equation solved
[...дополнительная информация]
p =
    1.0280    0.4186
ff =
   1.0e-09 *
   -0.0506   -0.4514
```

Решение систем нелинейных

$$\begin{cases} x^3 + \cos y - 2 = 0 \\ \sin x^2 + \ln y = 0 \end{cases}; J = \left(\frac{\partial F_i}{\partial \beta_j}\right) = \begin{pmatrix} 3x^2 & -\sin y \\ 2x\cos x^2 & y^{-1} \end{pmatrix}$$

Решение с якобианом

```
function fsolve_ex
opt = optimset(…
    'Display','iter', …
   'Jacobian', 'on', …
   'DerivativeCheck', 'on');
xy0=[1 1];
f = @(p) eq(p(1),p(2));
[xy d] = fsolve(f, xy0, opt)
end

function [F, J] = eq(x,y)
F = [x^3+cos(y)-2; …
    sin(x^2) + log(y)];
J = [3*x^2, -sin(y);…
    2*x*cos(x^2), 1./y];
end
```

```
>> fsolve_ex
_____________________________________________________________

   DerivativeCheck Information

Objective function derivatives:
Maximum relative difference between user-supplied
and finite-difference derivatives = 1.47737e-08.

DerivativeCheck successfully passed.
_____________________________________________________________


                                        Norm of      First-order   Trust-region
 Iteration  Func-count     f(x)          step         optimality    radius
     0          1        0.919395                        1.23             1
     1          2        0.550436       0.775422         3.09             1
     2          3       0.0196684       0.164206        0.387             1
     3          4       7.818e-05      0.0504667       0.0206             1
     4          5      1.05649e-09     0.00339965     7.61e-05             1
     5          6      2.06816e-19     1.27224e-05     1.06e-09             1

Equation solved.

fsolve completed because the vector of function values is near zero
as measured by the default value of the function tolerance, and
the problem appears regular as measured by the gradient.

<stopping criteria details>


xy =

    1.0280    0.4186
```