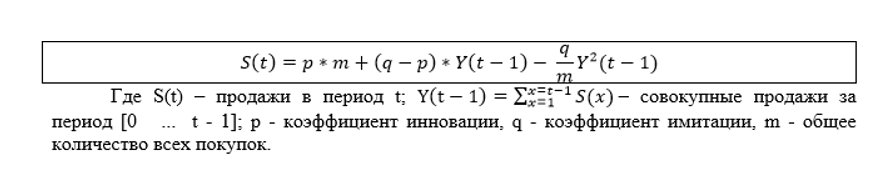
Выбор метода для определения параметров модели Басса прогнозирования развития возобновляемой энергетики на примере ветрогенерации на языке программирования Python с использованием библиотеки scipy

Слайд 2

* В качестве модели была использована формула Басса:



Суть модели Басса [4] заключается в том, что рост числа потребителей инновационного продукта объясняется двумя категориями:

• Те, кто в первую очередь пробуют новый продукт сами - инноваторы;

• Те, кто узнает о новом продукте из первой категории - имитаторы.

Слайд 3

Данные, используемые при исследовании

Были использованы данные по ветрогенерации за период с 1995 по 2020 гг в следующих регионах:

*Суммарные данные по миру*

*Европа*

*Северная Америка*

*Центральная и Южная Америка*

*Африка*

*Азиатско-Тихоокеанский регион*

*Средний Восток*

Слайд 4

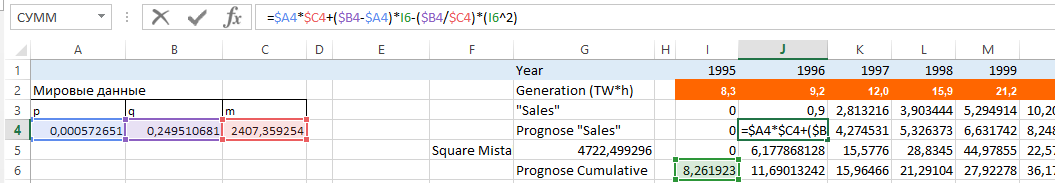
Методы применения модели к исследуемым данным

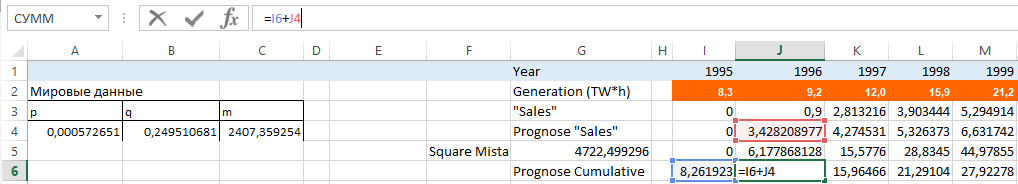
1. В ручном режиме, используя MS Excel
2. В автоматическом режиме, используя библиотеку Scipy языка программирования Python

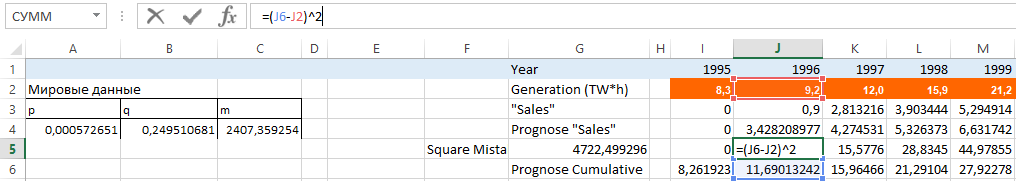
Проанализируем каждый из указанных методов

Слайд 5

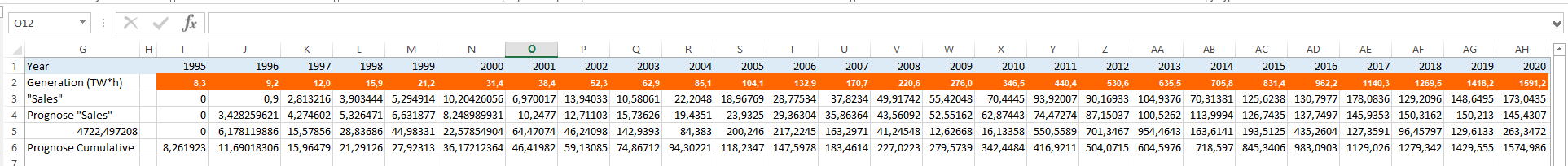
Исследование в ручном режиме

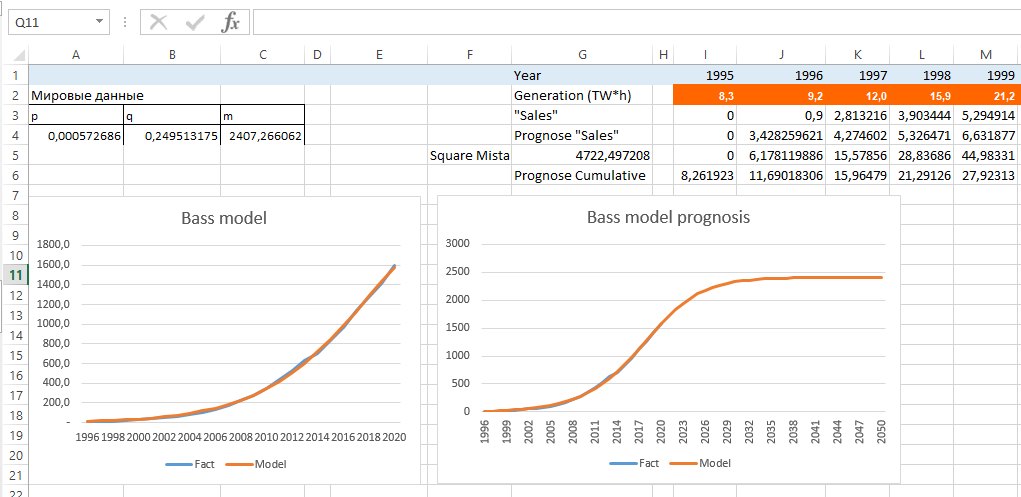






Слайд 6





Дополнительно, для работы данной модели, необходимо вручную подобрать начальные параметры p, q, m., подпираются они исключительно методом перебора, что бы данные приблизительно соответствовали фактическим, иначе оптимизация окажется неудачной!

Слайд 7

Автоматизация применения модели к исследуемым данным

Язык программирования Python, библиотека Scipy

def Bass(x, P, Q, M):

"""

Функция расчета Prognose Sales

x: величина Prognose Generation за прошлый год.

"""

return (P\*M+(Q-P)\*(x))-(Q/M)\*(x\*\*2)

Слайд 8

def squareMistake(k: tuple, \*sales) -> float:

"""

Функция для минимизации через scipy.

Рассчитывает сумму квадратов разностей значений Prognose Generation и Prognose Sales.

k: кортеж начальных параметров (P, Q, M);

sales: кортеж Sales.

"""

# Начальные значения для первого года

p0 = 0 # Prognose Sales

c0 = sales[0] # Prognose Generation

res = 0 # Значение суммы разности квадратов

# Набираем результат суммы разности квадратов за имеющиеся годы

for i in range(1, len(sales)):

p = Bass(c0, P=k[0], Q=k[1], M=k[2]) # Новый Prognose Sales

c = c0 + p # Новый Prognose Generation

res += (c - sales[i])\*\*2 # Добавляем к сумме разности квадратов

# Обновляем значения

p0 = p

c0 = c

return res

Слайд 9

Применение библиотеки Scipy

**scipy.optimize** - предоставляет функции для минимизации (или максимизации) целевых функций, возможно, с учетом ограничений. Он включает в себя решатели для нелинейных задач (с поддержкой алгоритмов локальной и глобальной оптимизации), линейного программирования, ограниченного и нелинейного метода наименьших квадратов, поиска корней и подбора кривой.

**curve\_fit** - функция, использующая нелинейный метод наименьших квадратов, чтобы подогнать функцию f к данным.

**minimize** – функция, минимизация скалярной функции одной или нескольких переменных.

Слайд 10

Подбор начальных значений p, q, m

popt, pcov = curve\_fit(Bass, data.generate, data.Sales,

bounds=(0, np.inf), method='trf', maxfev = 10000)

p0 = popt[0]

q0 = popt[1]

m0 = popt[2]

* popt - Оптимальные значения параметров, чтобы сумма разности квадратов f(xdata, \*popt) - ydata была минимизирована.
* pcov - Расчетная ковариация popt

Слайд 11

Минимизация суммы разности квадратов

k0 = [p0, q0, m0] # Начальные значения параметров

kb = ((0, None), (0, None), (0, None)) # Все параметры неотрицательные

res = minimize(squareMistake, k0, args=Generation, method=n, bounds=kb)

Returns:

x - массив решений,

success - логический флаг, указывающий, успешно ли завершился оптимизатор,

message - описывающее причину завершения.

n – метод, который применяет определенный алгоритм для оптимальной минимизации

Nelder-Mead - симплекс-метод является самым простым способом свести к минимуму явно определенную и довольно гладкую функцию

Powell - алгоритм минимизации без градиента

L-BFGS-B - метод от Бройдена–Флетчера–Гольдфарба–Шанно, реализованный с уменьшенным потреблением памяти за счет частичной загрузки векторов из матрицы Гессе

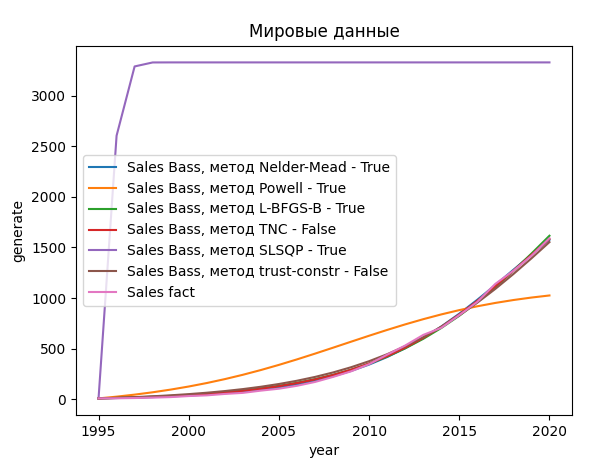
TNC - ограниченное число итераций, хорош для нелинейных функций с большим числом независимых переменных

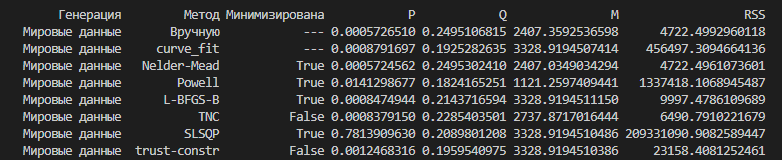
SLSQP - последовательное квадратичное программирование с ограничениями, ньютоновский метод решения системы Лагранжа

trust-constr - поиск локального минимума в доверительной области

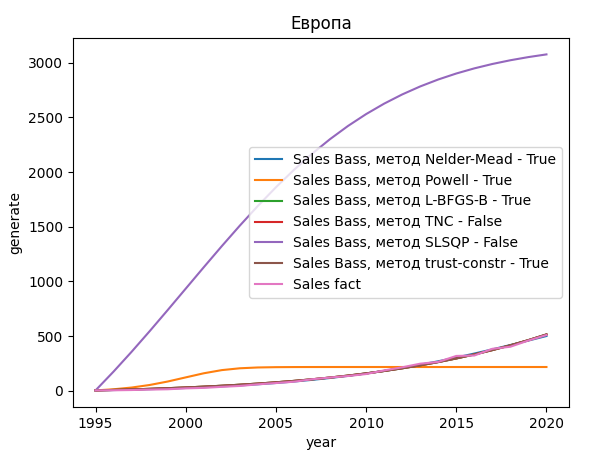
Слайд 12

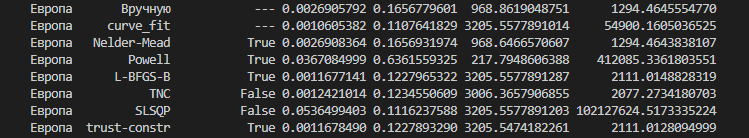
Поиск оптимального метода



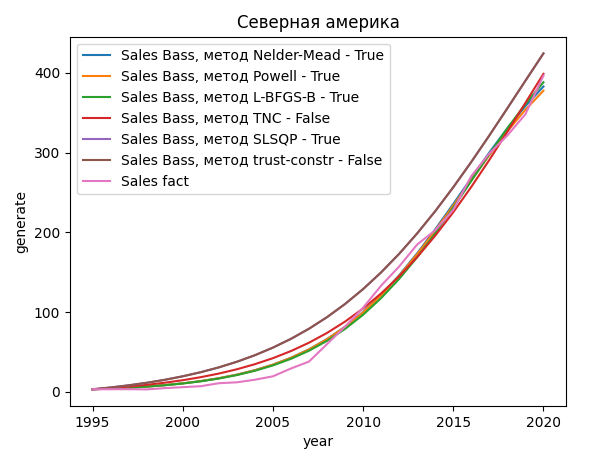


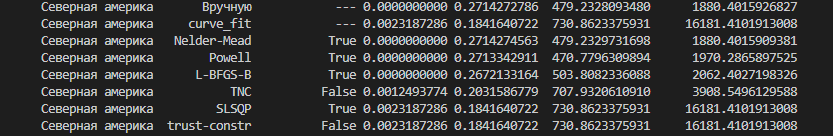
Слайд 13



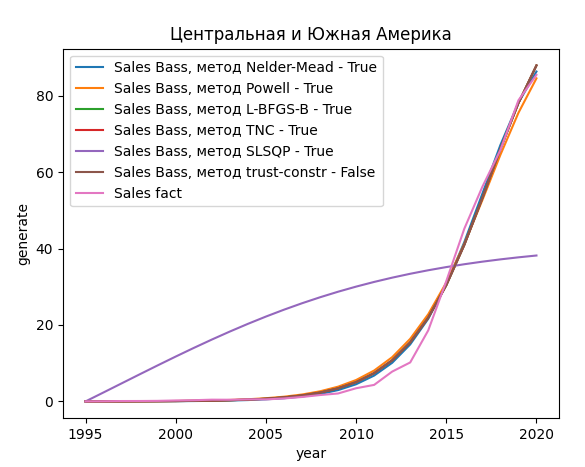


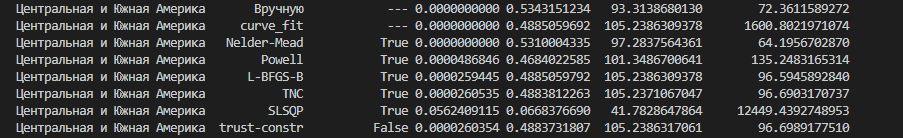
Слайд 14



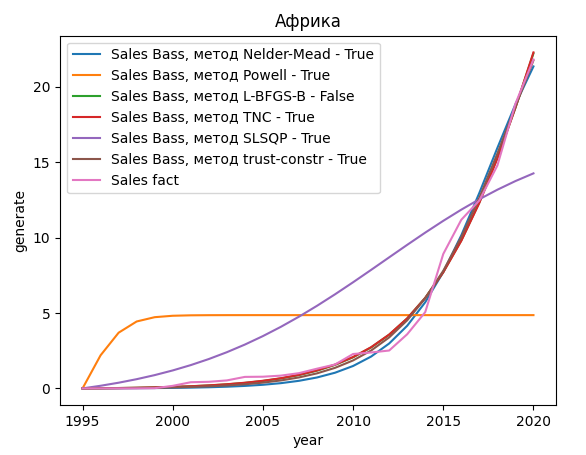


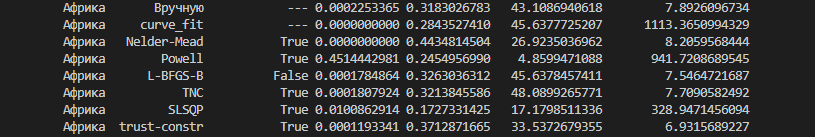
Слайд 15



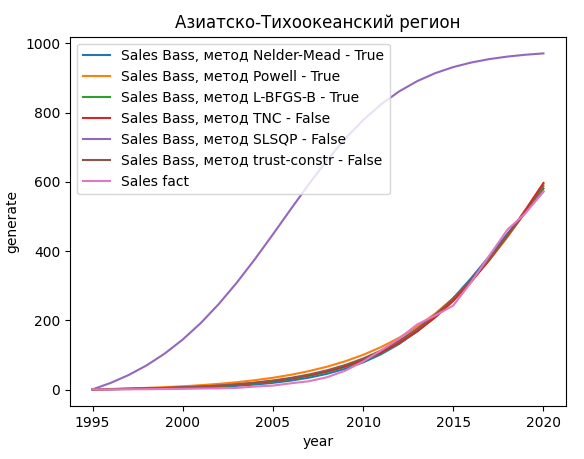


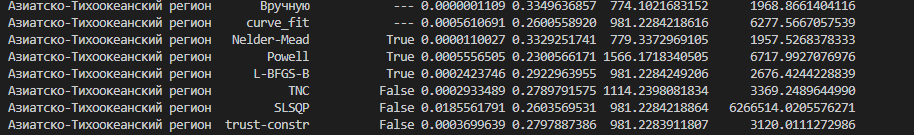
Слайд 16



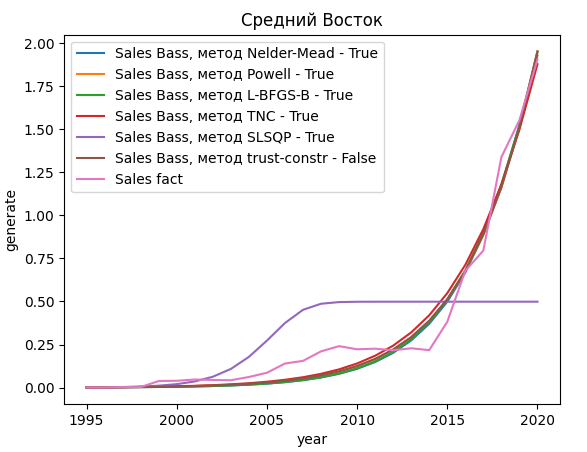


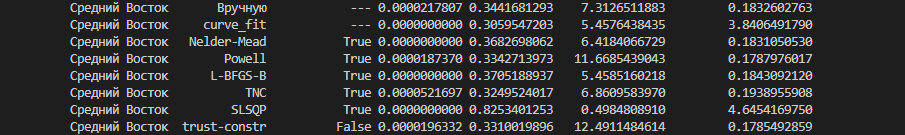
Слайд 17





Слайд 18





Слайд 19

**Заключение**

Проведя исследование по нахождению метода для определения параметров модели Басса на языке программирования Python с использованием библиотеки scipy, можно сделать вывод, что нет универсального алгоритма. Следовательно, необходимо использовать тот метод, который является самым оптимальным при подборе параметров.

В дальнейшей работе будет использоваться метод, который даёт наименьшую сумму разности квадратов.