# COMPUTER IMPLEMENTATION IN C++ LANGUAGE OF THE SIMPLEX METHOD FOR MINIMIZING FUNCTIONS IN MACHINE LEARNING PROBLEMS

Agamirov L.V.1,2, Agamirov V.L.1,2, Vestyak V.A.1, Frolova E.A.2

# Компьютерная реализация на языке С++ симплекс-метода минимизации функций в задачах Машинного обучения

Агамиров Л.В.1,2, Агамиров В.Л.1,2, Вестяк В.А.1, Фролова Е.А.2

*1) Московский авиационный институт (национальный исследовательский университет), Институт 3, кафедра 311, kaf311@yandex.ru*

*2) «Московский технический университет связи и информатики» (МТУСИ), факультет «Цифровая экономика и массовые коммуникации», кафедра «Бизнес-информатика»,* [*avl095@mail.ru*](mailto:avl095@mail.ru)

*1) Moscow Aviation Institute (national research university), Institute 3, department 311, kaf311@yandex.ru*

*2) “Moscow Technical University of Communications and Informatics” (MTUSI), Faculty of Digital Economy and Mass Communications, Department of Business Informatics, avl095@mail.ru*

**Абстракт**

Статья посвящена решению задач, связанных с вычислением функций нецентральных статистических распределений, оценки параметров нелинейных систем уравнений максимального правдоподобия симплекс методом Нелдера-Мида, выполнен обзор специализированных библиотек для решения поставленных задач, представлены необходимые практические рекомендации. Разработан алгоритм и программы для решения обратной задачи вычисления квантилей распределения, а также алгоритм и программы для реализации симплекс метода Нелдера-Мида с целью оценки параметров прогрессивно цензурированных выборок применительно к нормальному (логнормальному) распределению и распределению Вейбулла с конкретными примерами расчетов. Все программы представлены в открытом доступе, имеются соответствующие ссылки.

Непрерывное стремление к увеличению скоростных характеристик всех типов современных транспортных средств, включая военные и гражданские летательные аппараты, приводит к увеличению статических, динамических и тепловых нагрузок на элементы конструкции в процессе эксплуатации. Это вызывает необходимость, наряду с созданием новых конструкторских решений, материалов и технологических процессов производства, совершенствования статистических методов оценки параметров случайных величин, обеспечивающих надежность и ресурс ответственных деталей машин и элементов конструкции, физико-механические свойства которых подвержены значительному рассеянию. Развитие нейронных сетей и методов машинного обучения приобретают все большую актуальность в области решения сложных статистических задач, а в связи с этим необходимы соответствующие компьютерные программы, обеспечивающие эффективный поиск решений при наличии многих случайных переменных, что в свою очередь позволяет осуществлять глобальную оптимизацию инженерно-технических проектов.

Используемые компьютерные программы для этих целей являются частью встроенных библиотек, входящих в стандарт языков программирования высокого уровня таких как Python, R, Java, Golang и др. В то же время указанные языки высокого уровня как правило созданы на базе низкоуровневых языков программирования, таких как C, C++. В связи с этим, представляется актуальным реализация вышеуказанных задач непосредственно на базовых языках низкого уровня, что повышает гибкость, а главное быстродействие решений, имеющее особую важность, в том числе при обработке больших данных.

К числу статистических задач, рассматриваемых в настоящей работе и, не имеющих, аналитического решения относятся следующие:

**группа 1** - задачи, связанные с вычислением **обратных функций** сложных статистических распределений (прежде всего нецентральных), в которых необходимо вычислять точные значения квантилей распределения, соответствующих заданной вероятности. Разумеется, в этом случае необходимо располагать весьма точным алгоритмом вычисления самой функции распределения, чаще всего заданной в интегральной форме, а затем построить алгоритм вычисления с заданной погрешностью ее пределов интегрирования;

**группа 2** - задачи **оценки параметров** функций, формируемых в соответствии с методом максимального правдоподобия, одним из основных методов в теории оценивания. В этих задачах (прежде всего при наличии глубокого цензурирования выборки, что не позволяет использовать весьма эффективный для полной выборки метод наименьших квадратов [1, 2]), в которые, как правило, включены и задачи из пункта 1, функция правдоподобия представляет собой нелинейную систему уравнений, подлежащую минимизации, при этом в этих функциях, общем случае, возможно наличие нескольких локальных экстремумов, что предъявляет особые требования к заданию начальных приближений.

Рассмотрим некоторые статистические пакеты, предназначенные для решения вышеуказанных задач, при этом будем отдавать предпочтения тем проектам, в которых имеется открытый код на С++, *javascript или Python* (языки, достаточно легко переводимые на C++), с целью воспроизводимости кода на любой компьютерной платформе. Закрытые статистические пакеты, типа Matlab [3], Statistica [4], «Mathcad» и другие им подобные здесь не рассматриваются. Работы [5-8] посвящены наиболее распространенному применению обратных функций: моделированию случайных величин методом обратных функций, а также разработке аналитических приближений для обращения функций вероятностных распределений [9]. Многие исследования посвящены новым методам представления аналитических аппроксимаций для прямых и обратных функций сложных нецентральных статистических распределений [10].

Следует отдать должное открытым и доступным алгоритмам и программам прикладной статистики в рамках большого проекта Royal Statistical Society «Applied Statistics algorithms» [11], содержащего около 250 алгоритмов начиная с 1968 г. (проект завершился в 1997 г.), переведенных с языка Algol на Fortran, а затем, стараниями Джона Буркардта (John Burkardt) на С++ [12]. В то же время, как показал опыт авторов, в этих алгоритмах содержатся серьезные погрешности в части аппроксимаций нецентральных распределений, особенно при малых объемах наблюдений. Точные вычисления содержатся в уникальных статистических таблицах [13, 14], которые выполнены, по всей вероятности, методами численного интегрирования, но являются недоступными для анализа кода и использования в задачах компьютерного моделирования. Кроме того, приведенные в табличном виде дискретные процентные точки вызывают необходимость интерполяции или экстраполяции, что, в свою очередь, снижает точность расчетов. Очевидно, что использование таблиц является анахронизмом при современном уровне развития информационных технологий и может служить лишь для контроля точности численных расчетов, а также в учебных целях.

Следует также упомянуть статистические программы, встроенные в ныне самый популярный язык *Python* [15], которые базируются, применительно к задачам первой группы в части нецентральных распределений, на динамических библиотеках с закрытым кодом. Отметим также, что в отличие от *Python*, в стандарт C++ эти библиотеки пока не включены. Задачи второй группы решаются в *Python* с помощью комплекса программ *minimize.py*, код которых открыт и может быть переведен на C++, что будет представлено ниже в авторском варианте.

Самым мощным инструментом для решения статистических задач на языке С++ является библиотека «Boost» [16, 17], преимуществом которой является высокая точность и быстродействие расчетов статистических параметров самых сложных распределений. Аппроксимации, встроенные в специализированный язык *R* [18] уступают по точности функциям «Boost». Недостатком библиотеки «Boost» является ее огромный объем и сложность установки.

В связи с этим в настоящей работе предлагается модификация компонентов пакета «Boost», необходимая для решения статистических задач, объемом всего 3,73 Мб [19], без потери точности и быстродействия. Установка «Boost», в отличие от рекомендованной производителями следующая. Сначала необходимо установить на диск «C» пакет MSYS2 (https://www.msys2.org) для программирования на C и С++, аккуратно выполнив все указанные в руководстве шаги. Затем в созданный на диске каталог (c:\msys64\ucrt64) в папку «include» поместить модифицированный пакет «Boost» (<c:\msys64\ucrt64\include\boost\math>), в котором всего 6 директорий, необходимых для решения вышеуказанных задач (рисунок 1). Разумеется, при необходимости, модифицированный пакет может быть легко дополнен файлами из полной библиотеки.

|  |  |
| --- | --- |
|  |  |

Рисунок 1. Рисунок 2.

В переменные среды следует добавить путь <c:\msys64\ucrt64\bin> (последняя строка на рисунке 2). Указанная установка адаптирована для создания C++ программ из командной строки (g++, c++). При использовании Visual Studio модифицированный пакет <boost> записывается, например, в папку «c:\Program Files\Microsoft Visual Studio\2022\Community\VC\Tools\MSVC\14.31.31103\include\boost\». В программный код следует включить следующие заголовочные файлы и пространство имен:

#include <boost/math/distributions/normal.hpp>

#include <boost/math/distributions/students\_t.hpp>

#include <boost/math/distributions/chi\_squared.hpp>

#include <boost/math/distributions/fisher\_f.hpp>

#include <boost/math/distributions/non\_central\_t.hpp>

#include <boost/math/distributions/non\_central\_chi\_squared.hpp>

#include <boost/math/distributions/non\_central\_f.hpp>

#include <boost/math/distributions/binomial.hpp>

using namespace boost::math;

Обращение к функциям пакета «Boost» показано ниже (cdf-cumulative distribution function, ppf – percent point function (quantile), pdf – probability density function):

//#############Normal Distribution############################

double norm\_cdf(double x) {

normal\_distribution<>d(0,1);

return(cdf(d,x));

}

double norm\_ppf(double p) {

if(p<=0 || p>=1) return 0;

normal\_distribution<>d(0,1);

return(quantile(d,p));

}

double norm\_pdf(double x) {

normal\_distribution<>d(0,1);

return(pdf(d,x));

}

//#############Student Distribution############################

double t\_cdf(double x,double f) {

students\_t\_distribution<>d(f);

return(cdf(d,x));

}

double t\_ppf(double p,double f) {

if(p<=0 || p>=1) return 0;

students\_t\_distribution<>d(f);

return(quantile(d,p));

}

double t\_pdf(double x,double f) {

students\_t\_distribution<>d(f);

return(pdf(d,x));

}

//#############Chi-Squared Distribution############################

double chi\_cdf(double x,double f) {

chi\_squared\_distribution<>d(f);

return(cdf(d,x));

}

double chi\_ppf(double p,double f) {

if(p<=0 || p>=1) return 0;

chi\_squared\_distribution<>d(f);

return(quantile(d,p));

}

double chi\_pdf(double x,double f) {

chi\_squared\_distribution<>d(f);

return(pdf(d,x));

}

//#############F-Distribution############################

double f\_cdf(double x,double f1,double f2) {

fisher\_f\_distribution<>d(f1,f2);

return(cdf(d,x));

}

double f\_ppf(double p,double f1,double f2) {

if(p<=0 || p>=1) return 0;

fisher\_f\_distribution<>d(f1,f2);

return(quantile(d,p));

}

double f\_pdf(double x,double f1,double f2) {

fisher\_f\_distribution<>d(f1,f2);

return(pdf(d,x));

}

//#############Non Central t-Distribution############################

double nct\_cdf(double x,double f,double delta) {

non\_central\_t\_distribution<>d(f,delta);

return(cdf(d,x));

}

double nct\_ppf(double p,double f,double delta) {

if(p<=0 || p>=1) return 0;

non\_central\_t\_distribution<>d(f,delta);

return(quantile(d,p));

}

double nct\_pdf(double x,double f,double delta) {

non\_central\_t\_distribution<>d(f,delta);

return(pdf(d,x));

}

//#############Non Central chi-squared-Distribution############################

double nchi\_cdf(double x,double f,double delta) {

non\_central\_chi\_squared\_distribution<>d(f,delta);

return(cdf(d,x));

}

double nchi\_ppf(double p,double f,double delta) {

if(p<=0 || p>=1) return 0;

non\_central\_chi\_squared\_distribution<>d(f,delta);

return(quantile(d,p));

}

double nchi\_pdf(double x,double f,double delta) {

non\_central\_chi\_squared\_distribution<>d(f,delta);

return(pdf(d,x));

}

//#############Non Central F-Distribution############################

double ncf\_cdf(double x,double f1,double f2,double delta) {

non\_central\_f\_distribution<>d(f1,f2,delta);

return(cdf(d,x));

}

double ncf\_ppf(double p,double f1,double f2,double delta) {

if(p<=0 || p>=1) return 0;

non\_central\_f\_distribution<>d(f1,f2,delta);

return(quantile(d,p));

}

double ncf\_pdf(double x,double f1,double f2,double delta) {

non\_central\_f\_distribution<>d(f1,f2,delta);

return(pdf(d,x));

}

Для решения задач 1 группы необходимо отметить также пакет *jStat* [20], написанный на *Javascript.* Основные статистические функции переведены авторами на C++[21]. Преимуществом пакета является высокая точность, соизмеримая с функциями «Boost», замкнутость программного кода в одном файле, а недостатком отсутствие нецентральных распределений, за исключением нецентрального *t*-распределения, причем без процентных точке (только cdf и pdf). Остановимся на решении задач первой группы в части вычисления обратных функций для этого случая. Алгоритм решения обратной задачи, как в комплексе «Boost», так и в программах «Applied Statistics algorithms» заключается в нахождения корня функции F(x)=0, представляющей собой квадрат (или абсолютное значение) разности между заданным значением функции распределения и ее расчетным значением, вычисленным в точке варьируемой переменной «*x*». Указанный алгоритм реализован авторами в переводе [17] с открытым кодом. Принципиальным моментом здесь является применение для нахождения корней функции соответствующих алгоритмов [22, 23] и программ [24], (например, в [21] реализован алгоритм [22]). Не рекомендуется использовать из-за сильной потери быстродействия для решения задач первой группы Simplex – метод, о котором пойдет речь ниже для решения задач 2 группы.

Для решения задач второй группы рассмотрим один из популярных и наиболее эффективных методов поиска минимума функций многих переменных - симплекс-метод деформируемого многогранника (метода Нелдера-Мида) [25, 26] (отметим, что в *Python* предусмотрен более широкий выбор методов минимизации). Симплекс представляет собой набор точек, образующих многогранник, где каждая точка представляет собой набор значений параметров оптимизируемой функции. Идея метода заключается в том, чтобы изменять и перемещать симплекс в пространстве параметров, чтобы найти оптимальное значение функции. Впервые компьютерная реализация метода деформируемого многогранника была представлена в работе [27] на языке *Fortran*. Современный вариант алгоритма реализован на языке Python в библиотеке SciPy [28]. В данной работе предлагается авторская реализация алгоритма на C++ [29].

Рассмотрим применение метода Нелдера-Мида для оценки параметров прогрессивно цензурированных выборок применительно к нормальному (логнормальному) распределению и распределению Вейбулла. В соответствии с методом максимального правдоподобия (ММП) [1] оценки параметров непрерывной не менее двух раз дифференцируемой функции распределения случайной величины в общем случае прогрессивно цензурированной выборки [30, 31] определяются решением системы уравнений максимального правдоподобия. Оценки максимального правдоподобия (ММП-оценки) определяются в точках экстремума функции максимального правдоподобия:

, (1)

где

- число наблюдений (число объектов, достигших критического состояния);



- число значений случайной величины, в которых наблюдаются объекты, не достигшие критического состояния;



- количество объектов, снятых с испытаний при достижении значения ;



- общее число испытанных объектов испытания;



 - значения, при которых наблюдаются не достигшие критического состояния объекты.

Оценки параметров функции распределения  определяются решением следующей системы уравнений размерности относительно ММП-оценок параметров распределения:



 , (2)

где производные плотности распределения  и функции распределения  по параметрам определяют конкретный вид системы уравнений (2) для того или иного закона распределения.

Оценки параметров  двухпараметрического логарифмически нормального распределения в общем случае прогрессивно цензурированной выборки определяются в соответствии с (2) как корни системы двух уравнений:

, (3)



, (4)



где

,

.

Компьютерная запись минимизируемой функции на C++, показана ниже:

//######################################################

struct ne\_simp {

int n;

vector <double>x;

vector<int>r;

};

ne\_simp nesm;

double NormalMinFunction(vector<double>xsimpl) {

double s1,s2,s3,s4,z,psi,p,d,c1,c2;

int i,kx;

s1 = 0; s2 = 0; s3 = 0; s4 = 0; kx = 0;

if (xsimpl[0]<=0) return 10000;

if (xsimpl[1]<=0) return 10000;

for (i=0;i<nesm.n;i++) {

z = (nesm.x[i]-xsimpl[0])/xsimpl[1];

d = normal\_pdf(z,0,1);

p = normal\_cdf(z,0,1);

psi = d / (1. - p);

s1 +=(1.-nesm.r[i])\*(nesm.x[i]-xsimpl[0]);

s2 += (1.-nesm.r[i])\*pow(nesm.x[i]-xsimpl[0],2);

s3 += nesm.r[i]\*psi;

s4 += nesm.r[i]\*psi\*z;

kx+=1-nesm.r[i];

}

c1=s1+xsimpl[1]\*s3;

c2=s2+pow(xsimpl[1],2)\*(s4-kx);

z=c1\*c1+c2\*c2;

return z;

}

Для двухпараметрического распределение Вейбулла с плотностью:

 (5)

и функцией распределения

 (6)

ММП-оценка параметра *b* рассчитывается как корень уравнения:

 , (7)

после чего оценку параметра определяют из уравнения:



. (8)

Компьютерная запись минимизируемой функции на C++, показана ниже:

//#################MLE Weibull Minimized Function#######################

double WeibullMinFunction(vector<double>xsimpl) {

double s1,s2,s3,z,b,c;

int i,k;

if (xsimpl[0]<=0) return(10000000.);

s1=0;s2=0;s3=0;k=0;

b=xsimpl[0];

for(i=0;i<nesm.n;i++) {

k+=(1-nesm.r[i]);

s1+=pow(nesm.x[i],b);

}

c=s1/k;

for(i=0;i<nesm.n;i++) {

z=(pow(nesm.x[i],b))/c;

s3+=z\*log(z);

s2+=(1-nesm.r[i])\*log(z);

}

c=s3-s2-k;

return c\*c;

}

Полный текст программы можно получить по ссылке [32]. Проверка моделей проведена на примере цензурированных выборок.

Ниже представлены исходные данные для нормального закона (файл MLE\_Normal.inp). Цензурированные значения отмечены индексом 1.

Sample\_size

20

Data

4.673020907 4.741939078 4.788875116 4.821513528 4.850646235 4.870403905 4.911157609 4.925312091 4.962842681 4.980003372 5.06069784 5.089905111 5.127104798 5.152288344 5.184691431 5.214843848 5.243038049 5.285557309 5.344392274 5.507855872

Censorizes

1 0 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 1

Результаты расчета для нормального закона выводятся в файл MLE\_Normal.out. Значком «\*» отмечены оценки параметров по наблюдениям. Элементы ковариационной матрицы оценок - v11, v12, v21, v22, Q – относительная точность выхода, icount – число итераций.

Method:MLE\_Normal

n=20

X

4.67302 , 4.74194 , 4.78888 , 4.82151 , 4.85065 , 4.8704 , 4.91116 , 4.92531 , 4.96284 , 4.98 , 5.0607 , 5.08991 , 5.1271 , 5.15229 , 5.18469 , 5.21484 , 5.24304 , 5.28556 , 5.34439 , 5.50786 ,

r

1 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 ,

cp\*=5.046467451929

cko\*=0.187818804146

Q=0.000000000000

icount=58

cp=5.114510364218

cko=0.222206220261

v11=0.980489871463

v12=-0.001849887839

v21=-0.001849887839

v22=0.530571462580

Ниже представлены исходные данные для распределения Вейбулла (файл MLE\_Weibul.inp).

Sample\_size

20

Data

2980.4091 3330.3572 4947.7124 5320.5833 6235.7132

6713.0790 7330.5166 8231.1757 8774.0586 8821.4352

9099.0633 9630.0083 12068.2687 13032.4174 13125.3913

13158.1253 15128.9725 17312.1445 22853.1307 23826.4224

Censorizes

1 0 1 0 0 1 0 0 0 0 1 0 0 0 1 0 0 0 0 1

Результаты расчета для распределения Вейбулла выводятся в файл MLE\_Weibul.out.

Method:MLE\_Weibull

n=20

X

2980.41, 3330.36 , 4947.71 , 5320.58 , 6235.71 , 6713.08 , 7330.52 , 8231.18 , 8774.06 , 8821.44 , 9099.06 , 9630.01 , 12068.3 , 13032.4 , 13125.4 , 13158.1 , 15129 , 17312.1 , 22853.1 , 23826.4 ,

log(X)

7.99982, 8.11083 , 8.50668 , 8.57934 , 8.73805 , 8.81181 , 8.8998 , 9.01568 , 9.07955 , 9.08494 , 9.11593 , 9.17264 , 9.39833 , 9.4752 , 9.4823 , 9.48479 , 9.62437 , 9.75916 , 10.0368 , 10.0786 ,

r

1 , 0 , 1 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 ,

cp\*=9.175681378458

cko\*=0.505237409662

Q=0.000000000000

icount=27

b=2.123145196380

c=14415.311326338986

aw=9.576046207227

sw=0.470999346491

v11=1.446360189725

v12=-0.118617570398

v21=-0.118617570398

v22=0.790956035963

Сравнение программы на C++ и *Python* проводилось по критериям быстродействия и количества итераций для достижения заданной точности. Количество итераций ограничивалось числом 500, точность на входе задавалась 10-15. Анализ результата показал, что по количеству итераций и достижению заданной точности программы практически идентичны, а по быстродействие программа на C++ работает на порядок быстрее.

Выводы

1. В статье рассмотрены задачи, связанные с вычислением функций нецентральных статистических распределений и задачи оценки параметров нелинейных систем уравнений симплекс методом Нелдера-Мида, формируемых в соответствии с методом максимального правдоподобия при наличии цензурирования выборки, а также выполнен обзор статистических библиотек для решения поставленных задач, представлены необходимые практические рекомендации, существенно упрощающие установку необходимых пакетов.

2. Разработан алгоритм и программы на языке C++ для решения обратной задачи точного нахождения квантилей сложных статистических распределений.

3. Разработан алгоритм и программы на языке C++ для реализации симплекс метода Нелдера-Мида с целью оценки параметров прогрессивно цензурированных выборок применительно к нормальному (логнормальному) распределению и распределению Вейбулла с конкретными примерами расчетов.

4. Все программы представлены в открытом доступе по соответствующим ссылкам.

Список литературы

1. Дж. Кендалл, А. Стьюарт. Статистические выводы и связи. М: «Наука»,1973, с. 899.
2. Агамиров Л.В. Методы статистического анализа механических испытаний: М: Интермет Инжиниринг, ISBN 5-89594-105-2, с.128,2004.
3. Matlab, <https://www.mathworks.com/products/matlab.html>).
4. Statistica (<https://www.statistica.com>).
5. A.B. Холкина. Моделирование случайных величин методом обратных функций. Материалы VII Международной молодежной научной конференции, «Математическое и программное обеспечение информационных, технических и экономических систем», Томск, 23-25 мая 2019, Томск Издательский Дом Томского государственного университета 2019, с.73-75.
6. A. Gil, J. Segura, and N. M. Temme. On the computation and inversion of the cumulative noncentral beta distribution. Appl. Math. Comput., 361:74–86, 2019.
7. A. Gil, J. Segura, and N. M. Temme. A new asymptotic representation and inversion method for the Student’s t distribution. Integral Transforms Spec. Funct., 33(8):597–608, 2022.
8. Агамиров Л.В., Вестяк В.А. [Программа вычисления обратных функций сложных статистических распределений](https://www.elibrary.ru/item.asp?id=47992272). Свидетельство о регистрации программы для ЭВМ 2022612358, 10.02.2022, Заявка № 2021680616 от 13.12.2021.
9. Попов Г.А. Формула обращения для рациональных характеристических функций вероятностных распределений. Вестник АГТУ,2018, № 2 (66), DOI: 10.24143/1812-9498-2018-2-7-22.
10. Amparo Gil, Javier Segura, Nico M. Temme. New asymptotic representations of the noncentral t-distribution, 2023, https://doi.org/10.48550/arXiv.2306.06681.
11. Applied Statistics algorithms (<http://lib.stat.cmu.edu/apstat/>).
12. https://people.sc.fsu.edu/~jburkardt/cpp\_src/cpp\_src.html.
13. Pearson E.S., Hartley H.O. Biometrica tables for statisticians, Cambridge University Press, 1972, p. 416.
14. Большев Л.Н., Смирнов Н.В. Таблицы математической статистики, 1988, Изд. Наука, с. 416.
15. Python*,* https://www.python.org.
16. Boost, <https://www.boost.org>.
17. А. Полухин. Разработка приложений на С++ с использованием Boost. М.: ДМК Пресс, 2020, 346 с.
18. The R Project for Statistical Computing, <https://www.r-project.org>.
19. boost.zip, 2024
20. jStat, JavaScript Statistical Library, https://github.com/jstat/jstat.git
21. distribution.cpp
22. J.H. Wegstein. Accelerating convergence of iterative processes. Communications of the ACM, 1(6):9-13, 1958.
23. Mueller, David E., A Method for Solving Algebraic Equations Using an Automatic Computer, MTAC,10 (1956), 208-215.
24. 1130 Scientific Subroutine Package Programmer's Manual Program Number 1130-CM-02X. IBM Corporation 112 East Post Road White Plains, N. Y. 10601, p. 191.
25. J. A. Nelder, R. Mead, A Simplex Method for Function Minimization. The Computer Journal, Volume 7, Issue 4, January 1965, Pages 308-313, <https://doi.org/10.1093/comjnl/7.4.308>
26. Цыбанов В.В. Программа минимизации функции многих переменных методом деформируемого многогранника (по Нелдеру и Миду), DOI:[10.13140/RG.2.2.31221.88803](http://dx.doi.org/10.13140/RG.2.2.31221.88803), 2017/12/07.
27. Д. Химмельблау. "Прикладное нелинейное программирование" Изд-во "Мир" 1975г.
28. Метод Нелдера-Мида в Scipy (<https://github.com/AVL095/Nelder-Mead_Python>).
29. Метод Нелдера-Мида на C++ (<https://github.com/AVL095/Nelder-Mead_C>).
30. А. С. Cohen. Progressively Censored Sampling in the Three Parameter Log-Normal Distribution. Technometrics, vol. 18, № 1, 1976, pp. 99-103.
31. А.С. Соhen. Multi-Censored Sampling in the Three Parameter Weibull Distribution. Technometrics, vol 17, № 3, 1975, pp. 347-350.
32. Simplex.zip