НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЕ УЧРЕЖДЕНИЕ
ИНСТИТУТ ЯДЕРНОЙ ФИЗИКИ им. Г.И.Будкера СО РАН

А.Д. Букин

ПРОГРАММА ЧИСЛЕННОЙ МИНИМИЗАЦИИ ФУНКЦИИ МНОГИХ ПАРАМЕТРОВ

ИЯФ 2004-78

НОВОСИБИРСК
2004

Программа численной минимизации функции многих параметров

А.Д. Букин

Институт ядерной физики им. Г.И. Будкера
630090 Новосибирск, СО РАН

Аннотация

В программе реализованы алгоритмы: метод «золотого сечения» для минимизации функции одной переменной, симплекс-метод, метод переменной метрики пространства, метод сопряжённых направлений, метод Ньютона, метод проб и ошибок, случайный поиск. Возможна комбинация разных методов. Для выбранной комбинации методов возможна процедура многократной минимизации из разных начальных точек. Программа написана на языке Фортран-77. Общий объём кода около 3000 строк.

Subroutine for numerical minimization of the function of many parameters

A.D. Bukin

Budker Institute of Nuclear Physics,
630090, Novosibirsk, Russia

Abstract

There are several algorithms of minimization implemented in this program: "Golden section" method for one parameter minimization, simplex method, variable metrics method, conjugate directions method, Newton's method, "try and fail" method, random search. It is possible to combine different methods. For the chosen combination of methods there is possible a multiple minimization procedure from different start points. Program language is Fortran 77. Total size of source code is about 3000 lines.

1 Введение

В экспериментальной физике элементарных частиц для минимизации функций многих переменных широко используется программа MINUIT [1], разработанная в ЦЕРН. Автор неоднократно пытался разработать эффективный алгоритм минимизации, начиная ещё с того времени, когда библиотеки программ из зарубежных ведущих центров по физике высоких энергий были недоступны в России. Однако, все попытки были безуспешны, и метод переменной метрики пространства, реализованный в программе MINUIT, является, по-видимому, наиболее мощным алгоритмом минимизации гладких функций.

Тем не менее, программа MINUIT не всесильна, и иногда возникают сомнения в правильности результата (конечно, это бывает, как правило, со сложными функциями). Кроме того, иногда может возникнуть необходимость провести минимизацию функции на вычислительной машине, где ещё не установлена библиотека программ ЦЕРНа, но есть компилятор Фортрана. Могут быть и другие причины использовать альтернативную программу минимизации. Для этих целей разработана компактная программа на языке Фортран-77, включающая в себя несколько перспективных алгоритмов, с возможностью произвольного сочетания этих алгоритмов, а также с возможностью многократного повторения минимизации для выбранного комплекта алгоритмов из разных начальных точек. При обработке экспериментальных данных со Сферического Нейтрального Детектора (СНД) [2] эта программа многократно использовалась для получения надёжных результатов.

Данная работа является и подробным описанием указанной программы, и сравнительным анализом качества её работы (в сравнении с MINUIT) на примере нескольких существенно различных функций.

2 Структура программы

2.1 Вызов программы

Вызов программы оформлен в простейшем виде. По сравнению с MINUIT, основное функциональное упрощение — невозможность использования аналитически вычисляемого градиента в тех алгоритмах, где он

требуется. Случаи, когда возможно вычислить градиент аналитически, достаточно редки, и для них можно написать отдельную версию программы.

Вызов:

```
call BUKMIN(FCN,Npar,x,iFlMin,dfm,Ncal,Fmin)
integer *4 Npar, Ncal, iFlMin(Npar)
real *8 x(Npar), dfm, Fmin
external FCN
```

Минимизируемая функция F в точке x N-мерного пространства должна вычисляться программой FCN при вызове

```
call FCN(N,F,x)
integer *4 N
real *8 F,x(N)
```

Входные параметры:

FCN — идентификатор подпрограммы, вычисляющей функцию.

Npar — количество параметров, от которых зависит функция.

x — массив, содержащий начальную точку для минимизации.

iFlMin — массив, содержащий признаки оптимизируемых параметров. Если iFlMin(k) = 0, то минимизация по этому параметру не производится — он остаётся с первоначально определённым значением.

dfm — требуемая точность минимизации — оценка найденного минимума не должна отличаться от истинного минимума больше, чем на эту константу.

Ncal — переменная, содержащая максимально допустимое количество вызовов программы FCN.

Выходные параметры:

x — массив с координатами точки минимума.

Ncal — возвращается реальное количество вызовов FCN, произведённое при поиске минимума.

Fmin — минимальное значение функции.

2.2 Общие блоки

Общие блоки в большой степени определяют работу программы. Кроме того, полный список названий общих блоков удобно иметь для анализа возможных коллизий в больших программах по названиям модулей. Для уменьшения вероятности совпадения имён в данной программе с именами других модулей пользователя имена общих блоков и программ выбраны начинающимися с bukmin.

Так как в фортране нет динамического распределения памяти, то максимально возможное количество параметров функции определяется в тексте программы в самом начале в переменной NparMax. Для изменения максимального числа параметров следует изменить значение этой константы в строке

parameter (NparMax=...

Краткое описание общих блоков:

common/bukmin00/Istr,Ivmm,Icgm,Isim,Ideb,Inew,Itaf,Iran

integer *4 Istr, Ivmm, Icgm, Isim, Ideb, Inew, Itaf, Iran

Istr — стратегия минимизации:

0 — простейшая стратегия. В каждом из выбранных методов проверяются минимальные критерии достижения минимума. Если в цепочке программ минимизации какая-то программа сообщает о достижении минимума, то минимизация на этом заканчивается. Многократных заходов минимизации из разных начальных точек не производится.

1 — в каждом из выбранных методов применяются более надёжные критерии достижения минимума. Если в цепочке программ какая-то программа сообщает о достижении минимума (кроме методов с ненадёжным критерием достижения минимума: симплекс-метода, метода проб и ошибок, метода случайного поиска), то минимизация на этом заканчивается. Если этого не происходит, то минимизация проводится многократно из разных начальных точек с алгоритмом выбора этих точек таким же, как при Istr=2.

2 — Всё как в предыдущем пункте, но многократная минимизация из разных начальных точек обязательно проводится, игнорируя сообщения о достижении минимума от отдельных под-

программ минимизации (детали алгоритмов выбора начальных точек для следующего захода минимизации изложены в разд. 10).

3 — Проводится многократная минимизация из разных начальных точек, игнорируя сообщения от отдельных подпрограмм минимизации о достижении минимума. Алгоритмы выбора каждой следующей начальной точки и критерий достижения минимума для Istr=2 и Istr=3 различаются.

Ivmm — если больше нуля, то метод переменной метрики пространства используется при минимизации. Очерёдность вызова программ минимизации с разными алгоритмами определяется значениями этих сигнальных переменных. Если Ivmm < Isim, то вначале используется метод переменной метрики пространства, а затем симплекс-метод. При равных значениях сигнальных переменных порядок применения методов минимизации следующий: метод переменной метрики пространства, метод сопряжённых градиентов, симплекс-метод, метод Ньютона, метод проб и ошибок, метод случайного поиска.

Icgm — такая же сигнальная переменная для метода сопряжённых градиентов.

Isim — сигнальная переменная для симплекс-метода.

Inew — сигнальная переменная для метода Ньютона.

Itaf — сигнальная переменная для метода проб и ошибок.

Iran — сигнальная переменная для метода случайного поиска.

По умолчанию установлены
Istr=1,Ivmm=Icgm=Itaf=Iran=0, Inew=1,Isim=2,Ideb=0.

Ideb — уровень отладочной печати. Каждый следующий уровень включает в себя все предыдущие.

0 — нет печати.

1 — начало и окончание каждого захода минимизации. Фиксация каждого нового минимального значения функции.

2 — значения градиентов, параметры и достижения каждого логического шага.

3 — максимально подробная печать.

common/bukmin01/Npartot,Nfree,NcalTot,NcalMax,indfree(Nfree)

integer *4 Npartot, Nfree, NcalTot, NcalMax, indfree

Npartot — то же самое, что Npar — общее количество параметров.

Nfree — число свободных (оптимизируемых) параметров.

NcalTot — общее количество вызовов программы FCN.

NcalMax — максимально допустимое число вызовов программы FCN.

indfree — массив индексов оптимизируемых параметров в массиве x(Npar).

common/bukmin02/Fbest,xbest(Npar)

real *8 Fbest, xbest — текущее минимальное значение и точка минимума.

common/bukmin03/Nrun, NrunMax, iRets(Npar)

integer *4 Nrun, NrunMax, iRets

Nrun/NrunMax — текущее/максимальное число заходов минимизации из разных начальных точек. Если Nrun превышает NrunMax, то новые заходы могут быть, но информация о самых старых заходах теряется и Nrun=NrunMax.

iRets — массив кодов завершения заходов минимизации.

common/bukmin04/Frun(NrunMax),xRun(NparMax,NrunMax)

real *8 Frun, xRun

Frun — минимальные значения функции, достигнутые в результате данного захода минимизации.

xRun — точки, в которых заканчивался каждый заход минимизации.

common/bukmin05/xLin(Npar)

real *8 xLin — рабочий массив для программы bukminr0

common/bukmin06/xStart(Npar),eDir(Npar),xMin(Npar)

real *8 xStart,eDir,xMin — рабочие массивы для программы bukminr1

common/bukmin07/h(Npar)

real *8 h — массив текущих шагов для вычисления градиента

common/bukmin08/Vmm(Npar,Npar),grad1(Npar),grad2(Npar), dG(Npar),xw1(Npar),xw2(Npar), dX(Npar), Vdg(Npar),Amat(Npar,Npar), Fsimsim(Npar,Npar), xsimsim(Npar, Npar+1), xw3(Npar), xw4(Npar)

real *8 Vmm, grad1, grad2, dG, xw1, xw2, dX, Vdg, Amat, Fsimsim, xsimsim, xw3, xw4
Рабочие массивы для разных подпрограмм минимизации.

2.3 Подпрограммы

Все подпрограммы, кроме BUKMINR0, работают с векторами в сокращённом пространстве, состоящем из переменных параметров функции.

BUKMINR0(FCN,Npar,xCur,Fcur)

integer *4 Npar

real *8 xCur(Npar), Fcur

external FCN

Программа вычисляет значение функции Fcur в указанной точке xCur в пространстве оптимизируемых параметров. Эта же программа регистрирует минимальное значение функции, достигнутое в данном счёте.

BUKMINR1(FCN,Nfree,xStart,eDir,dfm,xMin,Fmin)

integer *4 Nfree

real *8 xStart(Nfree), eDir(Nfree), xMin(Nfree), dfm, Fmin

external FCN

Программа одномерной минимизации из начальной точки xStart вдоль направления eDir с требуемой точностью dfm. Предполагается, что в начальной точке значение функции известно и передаётся программе в переменной Fmin. В результате минимизации методом «золотого сечения» получается точка xMin с минимальным значением Fmin.

BUKMINR2(FCN,Nfree,iLev,x0,F0,grad,xwork)

integer *4 Nfree,iLev

real *8 x0(Nfree), F0, grad(Nfree), xwork(Nfree)

external FCN

Вычисление градиента grad(.) в точке x0(.). Если iLev=0, то вычисление градиента проводится без контроля точности, иначе оценивается точность вычисления, и если она хуже 1%, то шаг уменьшается вдвое и вычисление повторяется. F0 — значение функции в точке x0. xwork(.) — рабочий массив.

На выходе из программы устанавливается код возврата в iLev: 0 — градиент вычислен, 1 — невозможно вычислить градиент.

BUKMINR3(FCN,dfm,Nfree,iRet,xmin,Fmin,Vmm,grad1,grad2, dG,xw1,xw2,dX,Vdg)

integer *4 Nfree,iRet

real *8 dfm, xmin(Nfree), Fmin, Vmm(Nfree,Nfree), grad1(Nfree), grad2(Nfree), dG(Nfree), xw1(Nfree), xw2(Nfree), dX(Nfree), Vdg(Nfree)

Минимизация функции FCN Nfree параметров методом переменной метрики пространства. iRet — код возврата (0 — минимум найден, 1 — минимум не найден). xmin при обращении должен содержать начальную точку минимизации, по выходу будет содержать точку минимума (если найден). Fmin — значение функции в минимуме. Остальные массивы — рабочие.

BUKMINR4(FCN,Nfree,Nsim,iRet,xmin,Fmin,dfm,xwsim, Fsim,xsim,xc,esim)

integer *4 Nfree,Nsim,iRet

real *8 xmin(Nfree), Fmin, dfm, xwsim(Nfree), Fsim(Nsim), xsim(Nfree,Nsim), xc(Nfree), esim(Nfree,Nfree)

Минимизация функции FCN Nfree параметров симплекс-методом. В xmin при входе содержится начальная точка, при выходе — точка с минимальным значением функции Fmin. iRet=0, если минимум найден, iRet>0, если условие достижения минимума не выполнено.

BUKMINR5(FCN,dfm,Nfree,iRet,xmin,Fmin,pvec,evec,grcdm1, grcdm2,Cicdm,gicdm,pnorm,gpi1,gpi2,xw1,xw2, alphi,dXcdm)

integer *4 Nfree,iRet

real *8 dfm, xmin(Nfree), Fmin, pvec(Nfree,Nfree), evec(Nfree,Nfree), grcdm1(Nfree), grcdm2(Nfree), Cicdm(Nfree), gicdm(Nfree), pnorm(Nfree), gpi1(Nfree), gpi2(Nfree), xw1(Nfree), xw2(Nfree), alphi(Nfree), dXcdm(Nfree)

Минимизация функции FCN Nfree параметров методом сопряжённых направлений. iRet — код возврата (0 — минимум найден, 1 — минимум не найден). xmin при обращении должен содержать начальную точку минимизации, по выходу будет содержать точку минимума. Fmin — значение функции в минимуме. Остальные массивы — рабочие.

BUKMINR6(FCN,dfm,Nfr,iRet,xmin,Fmin,Amat,Awork,g,dx, xw1,xw2)

integer *4 Nfr, iRet

real *8 dfm, xmin(Nfr), Fmin, Amat(Nfr,Nfr), Awork(Nfr,Nfr), g(Nfr), dx(Nfr), xw1(Nfr), xw2(Nfr)

Минимизация функции FCN по Nfr переменным параметрам методом Ньютона.

BUKMINR7(FCN,dfm,Nfr,iRet,xmin,Fmin,xw1)

integer *4 Nfr, iRet

real *8 dfm, xmin(Nfr), Fmin, xw1(Nfr)

Минимизация функции FCN по Nfr переменным параметрам методом проб и ошибок.

BUKMINR8(FCN,dfm,Nfr,iRet,xmin,Fmin,xw1,dX,g)

integer *4 Nfr, iRet

real *8 dfm, xmin(Nfr), Fmin, xw1(Nfr), dX(Nfr), g(Nfr)

Минимизация функции FCN по Nfr переменным параметрам методом случайного поиска.

BUKMINR9(g)

real *8 g

Простой генератор нормальной случайной переменной.

3 Вычисление градиента

Для каждой координаты хранится текущее значение шага h_i для вычисления градиента. В начале счёта он полагается равным 10^{-7}, но в процессе счёта может изменяться. Ограничение снизу — 10^{-10} от абсолютного значения координаты. На значение самой компоненты градиента тоже накладывается ограничение — если значение превышает 10^{20}, то считается, градиента не существует.

Если iLev=0, то градиент вычисляется простейшим образом:

$$\frac{\partial F}{\partial x_i} \approx \frac{F(x_i + h_i) - F(x_i)}{h_i}.$$

Если $iLev \neq 0$, то градиент вычисляется, как минимум, по трём точкам:

$$g_i = \frac{\partial F}{\partial x_i} \approx \frac{F(x_i + h_i) - F(x_i - h_i)}{2h_i}.$$

При этом квадратичный член оценивается как

$$q_i = \frac{1}{2}\frac{\partial^2 F}{\partial x_i^2} \approx \frac{F(x_i + h_i) + F(x_i - h_i) - 2F(x_i)}{2h_i^2},$$

и шаг h_i достаточно мал, если

$$|g_i| \times 10^{-1} > |q_i \cdot h_i|.$$

Если данная компонента градиента близка к нулю, то это соотношение перестаёт выполняться при сколь угодно малом шаге. Тогда оценка градиента производится по 5 точкам: функция разлагается в ряд Тейлора

$$F(x) \approx F(x_i) + g_i \cdot \Delta x_i + q_i \cdot \Delta x_i^2 + c_i \cdot \Delta x_i^3 + d_i \cdot \Delta x_i^4,$$

где

$$g_i = \frac{\partial F}{\partial x_i} \approx \frac{8 \cdot \left[F\left(x_i + \frac{h_i}{2}\right) - F\left(x_i - \frac{h_i}{2}\right)\right] + F(x_i - h_i) - F(x_i + h_i)}{6h_i},$$

$$q_i = \frac{1}{2}\frac{\partial^2 F}{\partial x_i^2} \approx \frac{16 \cdot \left[F\left(x_i + \frac{h_i}{2}\right) + F\left(x_i - \frac{h_i}{2}\right)\right] - F(x_i - h_i) - F(x_i + h_i) - 30F(x_i)}{6h_i^2},$$

$$c_i = \frac{1}{6}\frac{\partial^3 F}{\partial x_i^3} \approx \frac{2 \cdot [F(x_i + h_i) - F(x_i - h_i)] - 4 \cdot \left[F\left(x_i + \frac{h_i}{2}\right) - F\left(x_i - \frac{h_i}{2}\right)\right]}{3h_i^3},$$

$$d_i = \frac{1}{24}\frac{\partial^4 F}{\partial x_i^4} \approx \frac{12F(x_i) + 2 \cdot [F(x_i + h_i) + F(x_i - h_i)] - 8 \cdot \left[F\left(x_i + \frac{h_i}{2}\right) + F\left(x_i - \frac{h_i}{2}\right)\right]}{3h_i^4}.$$

Здесь критерий достаточно малого шага h_i выбран в другом виде:

$$|q_i| \cdot 10^{-2} > |c_i|\, h_i + |d_i|\, h_i^2.$$

В любом случае, даже если этот критерий не выполняется, шаг h_i не уменьшается бесконечно — есть предел

$$h_i > \max\left\{10^{-10}; |x_i| \cdot 10^{-10}\right\}.$$

4 Метод переменной метрики пространства (VMM)

Известный метод переменной метрики пространства [3, 4] в данной программе реализован в следующем виде.

Вначале матрица метрики пространства полагается диагональной с $V_{ii} = 1$. Затем на каждом шаге минимизации, используя значения градиента в начале шага $\mathbf{g}_k$ и в конце шага $\mathbf{g}_{k+1}$, а также величину самого шага $\mathbf{\Delta x}_k$, производится преобразование матрицы.

Чтобы выбрать тип преобразования, вычисляются величины

$$\alpha_1 = \mathbf{\Delta x}^T \mathbf{\Delta g}, \quad \alpha_2 = \mathbf{\Delta g}^T \mathbf{V}_k \mathbf{\Delta g},$$

где

$$\mathbf{\Delta g} = \mathbf{g}_{k+1} - \mathbf{g}_k.$$

Если $\alpha_1/(\alpha_1 - \alpha_2) < 0$, то

$$\mathbf{V}_{k+1} = \mathbf{V}_k + \frac{\mathbf{\Delta x}\mathbf{\Delta x}^T}{\alpha_1} - \frac{\mathbf{V}_k\mathbf{\Delta g}\mathbf{\Delta g}^T\mathbf{V}_k}{\alpha_2}, \tag{1}$$

в противном случае

$$\mathbf{V}_{k+1} = \left(1 - \frac{\mathbf{\Delta x}\mathbf{\Delta g}^T}{\alpha_1}\right)\mathbf{V}_k\left(1 - \frac{\mathbf{\Delta g}\mathbf{\Delta x}^T}{\alpha_1}\right) + \frac{\mathbf{\Delta x}\mathbf{\Delta x}^T}{\alpha_1}. \tag{2}$$

Очередной шаг вычисляется по формуле

$$\mathbf{\Delta x} = -\mathbf{V}_k\mathbf{g}_k.$$

Если вычисленный таким образом шаг превышает 10, то он полагается равным 10. Первые n шагов и преобразований матрицы проводятся без проверки, уменьшается ли значение функции (n — размерность пространства). Но затем, если на очередном шаге функция не уменьшается, то в этом же направлении проводится линейная минимизация. Если число шагов преобразования матрицы больше n, очередной шаг не был чрезмерно большим, все диагональные элементы матрицы положительны и предсказание минимального значения функции

$$F_{min} = F_k - \frac{1}{2}\mathbf{g}_k^T\mathbf{V}_k\mathbf{g}_k$$

отличается от реального значения функции в предполагаемой точке минимума меньше, чем на $\Delta F_{min}/2$ — половину требуемой точности минимизации, то эта точка признаётся минимумом. Такой критерий достижения минимума используется при нулевой стратегии $Istr = 0$. При $Istr > 0$ дополнительно требуется, чтобы максимальное значение диагонального элемента матрицы, умноженное на квадрат длины градиента, было меньше $\Delta F_{min}/2$.

5 Метод сопряжённых направлений (CDM)

Метод реализован в соответствии со схемой из работы [5]. Как и в методе переменной метрики, все вычисления основаны на локальном представлении функции в виде квадратичной формы.

$$F(\mathbf{X}) = F_0 + \mathbf{B}^T\mathbf{X} + \frac{1}{2}\mathbf{X}^T\mathbf{A}\mathbf{X}. \tag{3}$$

В процессе минимизация происходит построение двух систем векторов $\mathbf{p}_i, \mathbf{e}_i,\ i = 1, 2, \ldots n$. Вектора $\mathbf{e}_i$ ортогональны векторам $\mathbf{p}_i$:

$$\mathbf{e}_i^T \mathbf{p}_j = |\mathbf{p}_i| \cdot \delta_{ij} \cdot \gamma_i,$$

где γ_i — некоторые ненулевые коэффициенты, кроме того,

$$\mathbf{e}_i = \mathbf{A}\mathbf{p}_i. \tag{4}$$

Теперь, если мы разложим перемещение из начальной точки по векторам $\mathbf{p}_i$:

$$\Delta\mathbf{X} = \sum_{i=1}^{n} \alpha_i \frac{\mathbf{p}_i}{|\mathbf{p}_i|},$$

то получим значение функции

$$\begin{aligned} F(\mathbf{X} + \Delta\mathbf{X}) = F(\mathbf{X}) + \mathbf{B}^T\Delta\mathbf{X} + \mathbf{X}^T\mathbf{A}\Delta\mathbf{X} + \tfrac{1}{2}\Delta\mathbf{X}^T\mathbf{A}\Delta\mathbf{X} = \\ = F(\mathbf{X}) + \sum_{i=1}^{n} \left[\mathbf{B}^T\mathbf{p}_i \tfrac{\alpha_i}{|\mathbf{p}_i|} + \mathbf{X}^T\mathbf{A}\mathbf{p}_i \tfrac{\alpha_i}{|\mathbf{p}_i|} + \tfrac{\alpha_i^2\gamma_i}{2|\mathbf{p}_i|} \right]. \end{aligned}$$

Видно, что переменные α_i разделяются, и минимизировать можно по каждой из них отдельно. Удобно определять значения α_i, приводящие к минимуму, используя условие равенства нулю градиента.

$$\mathbf{g}(\mathbf{X} + \Delta\mathbf{X}) = \mathbf{g}(\mathbf{X}) + \mathbf{A}\Delta\mathbf{X} = \mathbf{g}(\mathbf{X}) + \sum_{i=1}^{n} \frac{\alpha_i}{|\mathbf{p}_i|}\mathbf{e}_i = 0,$$

что даёт

$$\alpha_i = -\frac{1}{\gamma_i}\mathbf{p}_i^T\mathbf{g}(\mathbf{X}).$$

Свойства матрицы A можно определять, используя соотношение, что изменение градиента на перемещении $\mathbf{\Delta X}$ связано с матрицей $\mathbf{A}$ следующим образом

$$\mathbf{\Delta g} = \mathbf{A}\mathbf{\Delta X}$$

и для квадратичной формы не зависит от самой точки $\mathbf{X}$. Каждое очередное перемещение в n-мерном пространстве формируется как линейная комбинация векторов $\mathbf{p}_i$:

$$\mathbf{\Delta X} = \sum_{i=1}^{n} \alpha_i \frac{\mathbf{p}_i}{|\mathbf{p}_i|}.$$

Введём верхние индексы как обозначение номера шага, тогда получаем

$$\Delta\mathbf{g}^{(k)} = \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)} = \sum_{i=1}^{n} \frac{\alpha_i}{|\mathbf{p}_i|}\mathbf{A}\mathbf{p}_i = \sum_{i=1}^{n} \frac{\alpha_i}{|\mathbf{p}_i|}\mathbf{e}_i.$$

Отсюда

$$\gamma_i = \frac{1}{\alpha_i}\mathbf{p}_i^T \Delta\mathbf{g}^{(k)}.$$

Теперь можно предложить последовательность шагов, которая позволяет построить систему векторов и одновременно эффективно спускаться к минимуму, причём для функции — истинной квадратичной формы,— за $(n+1)$ шагов прийти точно в точку минимума. Пусть при подготовке k-ого шага в точке $\mathbf{X}_k$ вычисленное значение градиента равно $\mathbf{g}^{(k)}$ и уже имеются вектора $\mathbf{p}_i, \mathbf{e}_i,\ i = 1, 2, \ldots, k-1$. Для удобства вычислений потребуем, чтобы все вектора $\mathbf{p}_i$ были нормированы на единицу. Выберем новый вектор

$$\mathbf{p}_k \sim -\mathbf{g}^{(k)} + \sum_{i=1}^{k-1} \beta_i^{(k)} \mathbf{p}_i,$$

$$\beta_i^{(k)} = \frac{1}{\gamma_i}\mathbf{e}_i^T \mathbf{g}^{(k)},\ \ \gamma_i = \mathbf{e}_i^T \mathbf{p}_i.$$

Здесь γ_i выражают именно скалярное произведение векторов. Такой выбор коэффициентов β_i обеспечивает ортогональность $\mathbf{p}_i$ и $\mathbf{e}_j$ при $i \neq j$. Делаем новый шаг

$$\mathbf{\Delta X}_k = \sum_{i=1}^{n} \alpha_i^{(k)} \mathbf{p}_i,\ \ \alpha_i^{(k)} = -C_i^{(k)} \mathbf{p}_i^T \mathbf{g}^{(k)},$$

где константы $C_i^{(k)}$ вначале заполнены произвольными числами, например, $C_k^{(k)} = 0.1$. После этого делается переход в новую точку и в ней вычисляется новое значение градиента $\mathbf{g}^{(k+1)}$ и вектор изменения градиента

$$\Delta\mathbf{g}^{(k)} = \mathbf{g}^{(k+1)} - \mathbf{g}^{(k)}.$$

Это даёт возможность вычислить сопряжённый вектор

$$\mathbf{e}_k = \frac{1}{\alpha_k^{(k)}} \cdot \left(\Delta\mathbf{g}^{(k)} - \sum_{i=1}^{k-1} \alpha_i^{(k)} \mathbf{e}_i \right)$$

и правильное значение констант

$$C_i^{(k+1)} = \frac{\alpha_i}{\mathbf{p}_i^T \Delta \mathbf{g}^{(k)}}, \quad i = 1, 2, \ldots, k\,.$$

Последовательно выполнив n шагов, мы построим все пары сопряжённых векторов и следующим шагом попадём точно в минимум, если минимизируемая функция — квадратичная форма. Однако, для большинства функций на практике минимум не будет достигнут. Надо делать новые шаги, заменяя при этом какие-то из старых векторов. Проблема выбора «жертвы» не такая простая. В работе [5] предлагается следующий критерий, можно ли заменять вектор $\mathbf{p}_k$ на новый $\mathbf{p}_k^*$:

- Вычисляется новый вектор $\mathbf{p}_k^*$, используя значение градиента и вектора, кроме k-ого.
- Если $\mathbf{e}_k^T \mathbf{p}_k^* > \mathbf{e}_k^T \mathbf{p}_k$, замена производится, иначе делается попытка заменить другой вектор.

Эта процедура достаточно трудоёмкая, и нет гарантии, что можно будет заменить какой-нибудь вектор. В данной программе были проверены на примере функций 1, 2, 7 из таблицы 2 (у которых везде определён градиент) несколько более простых способов:

- Строго по очереди (циклически).
- Заменяется вектор с максимальным произведением $\mathbf{p}_k^T \mathbf{g}$.
- Заменяется вектор с минимальным произведением $\left|\mathbf{p}_k^T \mathbf{g}\right|$.
- Заменяется вектор с максимальным произведением $\left|\mathbf{e}_k^T \mathbf{g}\right| / \left|\mathbf{e}_k\right|$.
- Заменяется вектор, соответствующий максимальному отношению $\left|\mathbf{p}_k^T \mathbf{g}^{(k+1)} / \mathbf{p}_k^T \mathbf{g}^{(k)}\right|$.
- Заменяется вектор, соответствующий минимальному отношению $\mathbf{p}_k^T \mathbf{e}_k / \left|\mathbf{e}_k\right|$.
- После формирования полного набора векторов и проверки функции в точке с расчётными шагами по каждому вектору p_i, все вектора «забываются» и строятся заново.

Результат сравнения приведён в таблице 1. Предпочтительным представляется способ, когда после построения полной системы векторов делается

Таблица 1: Результаты минимизации (F_{min}/N_{cal}) функций 1, 2, 7 методом сопряжённых направлений при разных критериях выбора пары векторов $\mathbf{e}_i, \mathbf{p}_i$ для перевычисления.

Критерий замены	Номер функции		
	1	2	7
$\max \mathbf{p}_k^T \mathbf{g}$	$9.8 \cdot 10^{-5}/269$	$0.0079/53$	$342.1/10^6$
По очереди	$2.3 \cdot 10^{-5}/281$	$0.0079/53$	$303.2/1267$
$\min \left\lvert \mathbf{p}_k^T \mathbf{g} \right\rvert$	$2.0 \cdot 10^{-6}/1134$	$0.0011/148$	$15.20/52779$
$\max \left\lvert \mathbf{e}_k^T \mathbf{g} \right\rvert / \left\lvert \mathbf{e}_k \right\rvert$	$9.3 \cdot 10^{-5}/225$	$2.2 \cdot 10^{-9}/4746$	$0.598/10^6$
$\max \left\lvert \mathbf{p}_k^T \mathbf{g}^{(k+1)} / \mathbf{p}_k^T \mathbf{g}^{(k)} \right\rvert$	$1.1 \cdot 10^{-5}/229$	$2.2 \cdot 10^{-9}/4746$	$77.66/292758$
$\min \mathbf{p}_k^T \mathbf{e}_k / \left\lvert \mathbf{e}_k \right\rvert$	$0.00298/434$	$2.2 \cdot 10^{-9}/4746$	$0.158/152220$
$\mathbf{e}_k^T \mathbf{p}_k^* > \mathbf{e}_k^T \mathbf{p}_k$	$0.655/97$	$0.0186/29$	$421.8/273$
Всё заново	$4.6 \cdot 10^{-7}/199$	$8.0 \cdot 10^{-5}/64$	$0.130/793$

шаг в предполагаемый минимум, затем проверяются критерии достижения минимума, и если это не минимум, то построение системы векторов начинается заново. Далее используется именно эта версия программы.

Другой важный вопрос — когда прекращать минимизацию. Когда уже сформированы все n векторов, то последний шаг предполагает переход точно в минимум, причём изменение функции на этом шаге известно:

$$\Delta F_k = -\sum_{i=1}^{n} \frac{C_i \left(\mathbf{p}_i^T \mathbf{g}^{(k)}\right)^2}{2} \cdot \left|2 - \gamma_i C_i\right|.$$

Если реальное изменение функции отличается от этой оценки больше, чем на $\frac{1}{2}\Delta F_m$, минимум не достигнут. Если предсказание было удачным, то ещё проверяется, не слишком ли велико предсказание для следующего изменения функции. Если

$$\sum_{i=1}^{n} \frac{C_i \left(\mathbf{p}_i^T \mathbf{g}^{(k+1)}\right)^2}{2} < \frac{1}{10}\Delta F_m,$$

то минимум считается достигнутым. Дополнительно можно проверять, насколько самосогласованы C_i и γ_i. Для квадратичной формы должно выполняться соотношение $\gamma_i C_i = 1$. Чтобы проверять его в сравнении с требуемой точностью ΔF_m, можно потребовать

$$\sum_{i=1}^{n} \frac{C_i \left(\mathbf{p}_i^T \mathbf{g}^{(k)}\right)^2}{2} \cdot \left|1 - \gamma_i C_i\right| < \frac{1}{10}\Delta F_m.$$

6 Симплекс-метод

Симплекс-метод [4, 6] позволяет эффективно спускаться по крутому склону в область минимума, не требуя вычисления градиента, но скорость сходимости к точке минимума у него гораздо ниже, чем у метода переменной метрики пространства.

Вначале в n-мерном пространстве переменных параметров функции формируется набор из $(n+1)$ точки (начальная точка и точки с шагом H_0 вдоль каждой оси). Затем в цикле точка с самым большим значением функции заменяется на новую точку с меньшим значением. Выбор новой точки производится по следующему алгоритму.

1. Определяется номер k точки с самым большим значением функции $F(\mathbf{x}_k) \geq F(\mathbf{x}_i)$.

2. Вычисляется центр тяжести оставшихся точек:

$$\mathbf{x}_c = \frac{1}{N} \sum_{i \neq k} \mathbf{x}_i.$$

3. Вычисляется значение функции $F(\mathbf{x}_{ff})$ в точке

$$\mathbf{x}_{ff} = \mathbf{x}_c + 2 \cdot (\mathbf{x}_c - \mathbf{x}_k).$$

   Если $F(\mathbf{x}_{ff}) < F(\mathbf{x}_k)$, то точка $\mathbf{x}_k$ заменяется на $\mathbf{x}_{ff}$ и цикл повторяется с пункта 1.

4. Вычисляется значение функции $F(\mathbf{x}_f)$ в точке

$$\mathbf{x}_f = \mathbf{x}_c + (\mathbf{x}_c - \mathbf{x}_k).$$

   Если $F(\mathbf{x}_f) < F(\mathbf{x}_k)$, то точка $\mathbf{x}_k$ заменяется на $\mathbf{x}_f$ и цикл повторяется с пункта 1.

5. Вычисляется значение функции $F(\mathbf{x}_b)$ в точке

$$\mathbf{x}_b = \frac{1}{2} \cdot (\mathbf{x}_c + \mathbf{x}_k).$$

   По точкам $\mathbf{x}_k$, $\mathbf{x}_{ff}$, $\mathbf{x}_f$, $\mathbf{x}_b$ проводится парабола методом наименьших квадратов и в предполагаемой точке минимума $\mathbf{x}_m$

$$\mathbf{x}_m = \mathbf{x}_c + q \cdot (\mathbf{x}_c - \mathbf{x}_b), \quad q = \frac{F(\mathbf{x}_k) - F(\mathbf{x}_f)}{4a_2},$$

$$a_2 = \frac{52F(\mathbf{x}_{ff}) + 47F(\mathbf{x}_k) - 71F(\mathbf{x}_f) - 28F(\mathbf{x}_b)}{177}$$

тоже вычисляется функция $F(\mathbf{x}_m)$ (только в том случае, если $a_2 > 0$). Меньшее из двух значений функций $F(\mathbf{x}_m)$ и $F(\mathbf{x}_b)$ сравнивается с $F(\mathbf{x}_k)$. Если есть улучшение (функция уменьшается), то соответствующая точка заменяет $\mathbf{x}_k$ и цикл повторяется с пункта 1.

6. Если ни один из вариантов не подошёл, то начальный размер симплекса H_0 уменьшается в два раза и симплекс формируется заново. Но если H_0 уже меньше, чем 10^{-10}, то минимизация заканчивается с аварийным кодом возврата.

После каждой успешной замены самой плохой точки на точку с меньшим значением функции делается проверка на достижение минимума. Минимальное условие — разброс значений функции в симплексе меньше, чем величина $0.1\Delta F_m$. Если это выполняется, то при $Istr = 0$ минимизация прекращается.

При $Istr > 0$ делается дополнительная проверка, что точки симплекса не собрались в какое-то линейное подпространство (выйти из него симплекс «самостоятельно» не может). Для этого делается попытка построить ортогональный базис $\mathbf{e}_i$, используя вектора $\mathbf{x}_i$. Для этого полагается $\mathbf{e}_1 \sim \mathbf{x}_2 - \mathbf{x}_1$. Каждый следующий вектор вычисляется по следующей формуле:

$$\mathbf{e}_{i+1} \sim \mathbf{x}_{i+1} - \mathbf{x}_1 + \sum_{j=1}^{i} \lambda_j \mathbf{e}_j, \quad \lambda_j = \mathbf{e}_j \cdot \mathbf{x}_1 - \mathbf{e}_j \cdot \mathbf{x}_{i+1}$$

Каждый новый вектор $\mathbf{e}_i$, после того как определено его направление, нормируется на единицу. Если при попытке нормировать получается нулевая длина вектора и нормировка невозможна, то делается вывод, что все точки собрались в линейное подпространство, «натянутое» на уже полученные вектора базиса — скорей всего, в профиле функции имеется узкое ущелье. В этом случае H_0 уменьшается вдвое (если уже меньше 10^{-10}, то завершение минимизации с аварийным кодом возврата) и симплекс строится заново. Если же проверка завершилась нормально, то минимизация прекращается — минимум достигнут.

7 Метод Ньютона

Параметры квадратичной формы, аппроксимирующей функцию в некоторой окрестности текущей точки, можно получить через таблицу зна-

чений.

$$\Delta F = \sum_{i=1}^{n} g_i \Delta x_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} a_{ij} \Delta x_i \Delta x_j,$$

где a_{ij} — симметричная матрица. Делая поочерёдно по каждой координате шаг h в плюс и минус, можно определить коэффициенты g_i и a_{ii}:

$$g_i = \frac{\Delta F(\Delta x_i = h) - \Delta F(\Delta x_i = -h)}{2h}, \quad a_{ii} = \frac{\Delta F(\Delta x_i = h) + \Delta F(\Delta x_i = -h)}{h^2}.$$

Теперь, делая попарно шаги одновременно по двум координатам $i \neq j$, получим

$$a_{ij} = \frac{1}{h} \cdot \left[\frac{\Delta F(\Delta x_i = h, \Delta x_j = h)}{h} - g_i - g_j \right] - \frac{a_{ii} + a_{jj}}{2}.$$

Для такого восстановления параметров мы использовали $\frac{n(n+3)}{2}$ дополнительных значений функции, кроме значения в текущей точке. Если $\sum\limits_{i=1}^{n} g_i^2 > 0$, то текущая точка не может быть минимумом. Стационарную точку аппроксимирующей квадратичной формы (в ней градиент равен нулю) можно найти из системы n линейных уравнений

$$\sum_{j=1}^{n} a_{ij} \Delta x_j = -g_i, \quad i = 1, 2, \ldots, n,$$

которую можно решить любым способом, например, последовательным исключением неизвестных. Если в этой точке предполагаемого минимума значение функции окажется меньше, чем в исходной, то перемещаемся в эту точку и всё повторяем сначала. Если же значение функции окажется больше, то проводим линейную минимизацию вдоль направления на эту точку.

Удобство этого метода в том, что каждый следующий шаг делается независимо от предыдущей информации, поэтому вблизи точки минимума восстановление параметров квадратичной формы проводится точно, и сходимость вблизи минимума должна быть быстрой.

В качестве критерия достижения точки минимума с заданной точностью трудно что-нибудь здесь придумать, кроме совпадения предсказанного значения функции (по квадратичной форме) с действительным значением в новой точке.

8 Метод проб и ошибок

Метод основан на поочерёдных пробах вдоль каждой координаты с некоторым шагом h_i. Если значение функции в пробной точке меньше, чем в текущей, то делается переход в эту точку, шаг увеличивается в $\alpha = 1.3$ раз, и повторяется проба в этом же направлении. Когда по данному направлению нет уменьшения функции, то текущий шаг h_i умножается на $\beta = 0.5$, его знак меняется на противоположный, и процедура повторяется для следующей координаты.

В целях повышения устойчивости минимизации при наличии корреляции между параметрами функции, при $Istr > 0$ процедура дополняется пробами при попарном изменении координат: одновременно $x_i + h_i$ и $x_j \pm h_j$ для всех $j \neq i$.

Минимум считается достигнутым, если при n^2 пробах по всем координатам изменение функции по абсолютной величине не превышало заданную величину ΔF_m. При $Istr > 0$ дополнительно проверяется, что суммарное понижение текущего минимума функции не превысило $0.1 \cdot \Delta F_m$ за последние n^2 проверок всех координат.

Минимизация завершается аварийно, если какой-то шаг становится меньше 10^{-10}. В таблицах этот метод будем обозначать как $T\&F$ (Try and Fail).

9 Случайный поиск

Реализован следующий вариант случайного поиска. В процессе минимизации для выбора новой точки используются два параметра: вектор $\mathbf{v}$ и радиус ρ. В начальный момент $\mathbf{v} = 0$, $\rho = 1$. Координаты новой точки $\mathbf{x}^{(k+1)}$ получаются следующим образом:

$$x_i^{(k+1)} = x_i^{(k)} + v_i + \rho \cdot g_i, \quad i = 1, 2, \ldots, n,$$

где g_i — случайные переменные, распределённые по нормальному закону.

Если значение функции в новой точке меньше текущего минимума, то ρ умножается на 0.9, $\mathbf{v}$ меняет направление на $\left(x_i^{(k+1)} - x_i^{(k)}\right)$, а длина увеличивается на $\left|x_i^{(k+1)} - x_i^{(k)}\right|$. Естественно, делается переход в новую точку $x_i^{(k+1)}$, и всё повторяется.

Если же значение функции в пробной точке больше текущего минимума, то $\mathbf{v}$ умножается на 0.5. Радиус ρ умножается на 1.5, если он меньше $|\mathbf{v}|$, или на 0.8, если он больше $|\mathbf{v}|$.

Уменьшение радиуса меньше 10^{-5} блокируется. Если это произошло, минимизация прекращается аварийно.

Минимизация прекращается, если за n^2 проб функция уменьшилась меньше, чем на ΔF_m и при каждой пробе вариация функции меньше ΔF_m.

10 Многократное повторение заходов минимизации

Многократное повторение заходов минимизации из разных начальных точек возможно только при стратегии $Istr > 0$.

Начальная точка самого первого захода для всех стратегий указывается пользователем во входном параметре — массиве $xmin$. Для стратегий Istr=1,2 начальная точка второго захода выбирается симметричной относительно достигнутого минимума. Для третьего захода начальной точкой является симметричное отражение худшего из предыдущих двух минимумов относительно лучшего, но на фиксированном расстоянии от лучшего минимума ($\Delta \mathbf{x}^2 = 1$). Четвёртый заход начинается из точки, расположенной на прямой, соединяющей самый лучший минимум с наиболее удалённым минимумом и тоже на фиксированном расстоянии, равном 1. Если предыдущих точек минимума больше трёх, используется следующая более сложная система.

Пусть $\mathbf{R}_i$, F_i — векторы в n-мерном пространстве переменных параметров, в которых заканчивались предыдущие заходы минимизации, и значения функции в этих точках. F_{min} — абсолютный минимум. Введём весовые множители w_i:

$$w_i = \exp\left(F_{min} - F_i\right),$$

которые придают больший вес нижележащим точкам. Для начала определим основное направление «ущелья», минимизируя сумму квадратов отклонений

$$S_1 = \sum_{i=1}^{k} w_i \cdot \left(\mathbf{R}_0 + t_i \cdot \mathbf{v}_1 - \mathbf{R}_i\right)^2,$$

где $\mathbf{R}_0$ — точка на дне «ущелья», единичный вектор $\mathbf{v}_1$ — направление вдоль дна «ущелья». Легко получить оптимальное значение параметров t_i точек на этой прямой, ближайших к $\mathbf{R}_i$

$$t_i = \left(\mathbf{R}_i - \mathbf{R}_0\right) \cdot \mathbf{v}_1$$

и «центральную» точку

$$\mathbf{R}_0 = \frac{\sum\limits_{i=1}^{k} w_i \mathbf{R}_i}{\sum\limits_{i=1}^{k} w_i}.$$

Теперь можно выбрать оптимальный вектор $\mathbf{v}_1$ — это оказывается собственный вектор матрицы

$$\mathbf{M} = \sum_{i=1}^{k} w_i \left(\mathbf{R}_i - \mathbf{R}_0\right) \left(\mathbf{R}_i - \mathbf{R}_0\right)^T,$$

соответствующий наибольшему собственному значению λ_1. В принципе, сумму квадратов отклонений S_1 теперь можно записать в виде

$$\min S_1 = Tr\left(\mathbf{M}\right) - \lambda_1.$$

Теперь, при фиксированных параметрах $\mathbf{R}_0,\ \mathbf{v}_1,\ t_i$, посмотрим, как можно улучшить аппроксимацию кривой, описывающей дно ущелья, введя квадратичную нелинейность в виде

$$\mathbf{r}(t) = \mathbf{R}_0 + t \cdot \mathbf{v}_1 + \left(\mu_0 + \mu_1 t + \mu_2 t^2\right) \cdot \mathbf{v}_2,$$

где единичный вектор $\mathbf{v}_2$ ортогонален вектору $\mathbf{v}_1$. Новая сумма квадратов отклонений

$$S_2 = \sum_{i=1}^{k} w_i \cdot \left(\mathbf{R}_0 + t_i \cdot \mathbf{v}_1 + \left(\mu_0 + \mu_1 t_i + \mu_2 t_i^2\right) \cdot \mathbf{v}_2 - \mathbf{R}_i\right)^2$$

при $\mu_0 = \mu_1 = \mu_2 = 0$ равна $\min S_1$. Так как это — вторая поправка, и не хотелось бы усложнять решение без особой необходимости, выберем вектор $\mathbf{v}_2$ как собственный вектор той же матрицы $\mathbf{M}$, соответствующий следующему по величине собственному значению λ_2. Учитывая свойства ранее фиксированных параметров, можно преобразовать сумму к виду

$$S_2 = \mathrm{Tr}\left(\mathbf{M}\right) - \lambda_1 + \sum_{i=1}^{k} w_i \cdot \left(\mu_0 + \mu_1 t_i + \mu_2 t_i^2\right)^2 - 2\mu_2 \cdot \sum_{i=1}^{k} w_i t_i^2 \cdot \left[\left(\mathbf{R}_i - \mathbf{R}_0\right) \cdot \mathbf{v}_2\right].$$

Отсюда легко получаются соотношения

$$\mu_0 = -\mu_2 \cdot \frac{\sum\limits_{i=1}^{k} w_i t_i^2}{\sum\limits_{i=1}^{k} w_i} = -\mu_2 \cdot <t_i^2>, \quad \mu_1 = -\mu_2 \cdot \frac{\sum\limits_{i=1}^{k} w_i t_i^3}{\sum\limits_{i=1}^{k} w_i t_i^2} = -\mu_2 \cdot \frac{<t_i^3>}{<t_i^2>}$$

и тогда

$$\mu_2 = \frac{\sum\limits_{i=1}^{k} w_i t_i^2 \cdot [(\mathbf{R}_i - \mathbf{R}_0) \cdot \mathbf{v}_2]}{\left(< t_i^4 > - < t_i^2 >^2 - \frac{< t_i^3 >^2}{< t_i^2 >} \right) \cdot \sum\limits_{i=1}^{k} w_i}.$$

Вся эта деятельность основывается на ожидании, что точки минимумов растянуты вдоль некоторого «ущелья», и следующую начальную точку желательно выбрать поближе к ожидаемому минимуму вдоль этого ущелья. Параметр t является координатой вдоль дна этого ущелья, и каждой точке $\mathbf{R}_i$ соответствует значение t_i. Аппроксимируем значения функции квадратичной зависимостью $F(t) = F_0 + a_1 t + a_2 t^2$, выбрав значения $F_0,\ a_1,\ a_2$, минимизирующие сумму

$$S_3 = \sum_{i=1}^{k} w_i \left(F_0 + a_1 t_i + a_2 t_i^2 - F_i \right)^2.$$

Коэффициенты легко могут быть найдены:

$$a_1 = \frac{(\sigma_2 Y_1 - \sigma_3 Y_0)\,\sigma_2 + (\sigma_3 Y_2 - \sigma_4 Y_1)\,\sigma_0 + (\sigma_4 Y_0 - \sigma_2 Y_2)\,\sigma_1}{(\sigma_1\sigma_4 - \sigma_2\sigma_3)\,\sigma_1 + (\sigma_2^2 - \sigma_1\sigma_3)\,\sigma_2 + (\sigma_3^2 - \sigma_2\sigma_4)\,\sigma_0},$$

$$a_2 = \frac{\left(\sigma_2^2 - \sigma_1\sigma_3\right) Y_0 + (\sigma_0\sigma_3 - \sigma_1\sigma_2)\,Y_1 + \left(\sigma_1^2 - \sigma_0\sigma_2\right) Y_2}{(\sigma_1\sigma_4 - \sigma_2\sigma_3)\,\sigma_1 + (\sigma_2^2 - \sigma_1\sigma_3)\,\sigma_2 + (\sigma_3^2 - \sigma_2\sigma_4)\,\sigma_0},$$

где

$$\sigma_0 = \sum_{i=1}^{k} w_i,\ \ \sigma_1 = \sum_{i=1}^{k} w_i t_i,\ \ \sigma_2 = \sum_{i=1}^{k} w_i t_i^2,\ \ \sigma_3 = \sum_{i=1}^{k} w_i t_i^3,\ \ \sigma_4 = \sum_{i=1}^{k} w_i t_i^4,$$

$$Y_0 = \sum_{i=1}^{k} w_i F_i,\ \ Y_1 = \sum_{i=1}^{k} w_i t_i F_i,\ \ Y_2 = \sum_{i=1}^{k} w_i t_i^2 F_i.$$

Теперь может быть получена точка t_0 предполагаемого минимума вдоль «ущелья»:

$$t_0 = -\frac{a_1}{2a_2}.$$

Конечно, минимумом это будет, только если $a_2 > 0$. Для того чтобы ограничить величину скачков, потребуем, чтобы $|t_0| \le 10 \cdot \max\limits_i |t_i|$.

При $Istr = 1$ последовательность заходов минимизации прерывается, если какая-либо из заказанных программ минимизации (кроме

SIMPLEX, T&F, Random) сообщает о достижении минимума или последние N_R заходов имеют разброс значений минимумов F_i меньше заказанной точности ΔF_m.

При $Istr > 1$ сообщение о достижении минимума какой-либо из заказанных программ минимизации игнорируется. Последовательность заходов при стратегии Istr=2 прерывается, если последние N_R заходов имеют разброс значений минимумов F_i меньше заказанной точности ΔF_m и предсказанное изменение минимума при продлении заходов до бесконечности не больше ΔF_m. Это предсказание получается следующей процедурой. Значения F_i аппроксимируются функцией индекса i:

$$F_i \approx A + B \cdot q^i$$

так, что минимизируется сумма

$$S_3 = \sum_{i=1}^{k} w_i \cdot \left(A + B \cdot q^i - F_i\right)^2.$$

Минимизация по параметрам A и B выполняется аналитически:

$$A = \frac{Q_2 \cdot R_0 - Q_1 \cdot R_1}{Q_0 \cdot Q_2 - Q_1^2}, \quad B = \frac{Q_0 \cdot R_1 - Q_1 \cdot R_0}{Q_0 \cdot Q_2 - Q_1^2},$$

$$S_3 = \frac{\left(Q_2 \cdot R_2 - R_1^2\right) Q_0 - Q_1^2 R_2 + 2 Q_1 R_0 R_1 - Q_2 R_0^2}{Q_0 \cdot Q_2 - Q_1^2},$$

где

$$Q_0 = \sum_{i=1}^{k} w_i, \quad Q_1 = \sum_{i=1}^{k} w_i q^i, \quad Q_2 = \sum_{i=1}^{k} w_i q^{2i},$$

$$R_0 = \sum_{i=1}^{k} w_i F_i, \quad R_1 = \sum_{i=1}^{k} w_i q^i F_i, \quad R_2 = \sum_{i=1}^{k} w_i F_i^2.$$

По оставшемуся переменному параметру q минимизация проводится численно. Надо иметь в виду, что точки $q = 0$ и $q = 1$ являются особыми для функции S_3.

При q=0:

$$A = \frac{\sum\limits_{i=2}^{k} w_i F_i}{\sum\limits_{i=2}^{k} w_i}, \quad q \cdot B = \frac{\sum\limits_{i=2}^{k} w_i (F_1 - F_i)}{\sum\limits_{i=2}^{k} w_i},$$

$$S_3 = \frac{\left(\sum\limits_{i=2}^{k} w_i\right)\left(\sum\limits_{i=2}^{k} w_i F_i^2\right) - \left(\sum\limits_{i=2}^{k} w_i F_i\right)^2}{\sum\limits_{i=2}^{k} w_i}.$$

При q=1:

$$A = \frac{\sum\limits_{i=1}^{k} w_i F_i}{\sum\limits_{i=1}^{k} w_i}, \quad B = 0,$$

$$S_3 = \frac{\left(\sum\limits_{i=1}^{k} w_i\right)\left(\sum\limits_{i=1}^{k} w_i F_i^2\right) - \left(\sum\limits_{i=1}^{k} w_i F_i\right)^2}{\sum\limits_{i=1}^{k} w_i}.$$

Далее, минимизация из разных начальных точек продолжается, если $|q| \geq 1$, или $|A - F_{min}| > \Delta F_m$, или $S_3 > k\Delta F_m^2$.

При стратегии Istr=3 выбор начальной точки нового захода более осторожен и основывается только на координатах минимума последних двух заходов, и критерий достижения минимума более жёсткий — значения функций в последних двух минимумах должны отличаться не более, чем на ΔF_m, а расстояние между точками минимумов не должно превышать 10^{-10}. В выборе новой начальной точки, кроме векторов $\mathbf{x}_1$ и $\mathbf{x}_2$, указывающих на последние два минимума, используется переменный параметр h и число N_f безуспешных попыток подряд улучшить минимум. Предполагается, что $F_1 \geq F_2$. Тогда

$$\mathbf{x}_3^{(start)} = \mathbf{x}_2 + h \cdot \left[\frac{\mathbf{x}_2 - \mathbf{x}_1}{|\mathbf{x}_2 - \mathbf{x}_1| \cdot (1 + N_f)} + \frac{N_f}{1 + N_f} \cdot \mathbf{G}\right],$$

где $\mathbf{G}$ — случайный вектор, каждая компонента которого распределена по нормальному закону. По окончании минимизации получаем точку минимума $\mathbf{x}_3$ и значение функции в ней F_3. Теперь возможны три варианта:

- Самый благоприятный — $F_3 < F_2$. В этом случае полагаем $N_f = 0$, точку $\mathbf{x}_1$ «забываем», вторую точку переносим на место первой, а третью — на место второй. Если $|\mathbf{x}_3 - \mathbf{x}_2| < 0.001 \cdot h$, то $h \to h/2$, иначе $h \to 1.5h$.
- $F_1 > F_3 \geq F_2$. В этом случае $N_f \to N_f + 1$, $h \to h/2$. Первую точку «забываем», на её место переносим третью точку.

- Третий вариант — $F_3 \geq F_1 \geq F_2$. В этом случае третью точку «забываем», $N_f \to N_f + 1$, $h \to h/2$.

Такой алгоритм обязательно сходится к какой-нибудь точке, хотя и не гарантированно к точке минимума. С одной стороны, вероятность постепенного дрейфа вдоль дна ущелья в этом алгоритме повышенная, с другой стороны, даже для самых простых функций здесь будет не меньше, чем удвоение времени счёта.

11 Отладка программы

Для отладки программы и проверки эффективности алгоритмов были выбраны несколько тестовых функций, описанных в таблице 2. Далее везде будут ссылки на тестовые функции по их порядковому номеру в

Таблица 2: Тестовые функции для проверки эффективности алгоритмов минимизации. Минимальное значение функций равно нулю.

№	Кол. пар.	Функция	Точка минимума
1	5	$(1+x_1)^2 + 2^2 \cdot (2 + x_1 + 2^2 x_2)^2 + +3^2 \cdot (3 + x_1 + 2^3 x_2 + 3^3 x_3)^2 + +4^2 \cdot (4 + x_1 + 2^4 x_2 + 3^4 x_3 + 4^4 x_4)^2 + +5^2 \cdot (5 + x_1 + 2^5 x_2 + 3^5 x_3 + 4^5 x_4 + 5^5 x_5)^2$	(-1,-0.25,0,$\frac{1}{256}$,$-\frac{1}{3125}$)
2	2	$100 \cdot (x_2 - 0.01 \cdot x_1^2 + 1)^2 + 0.01 \cdot (x_1 + 10)^2$	(-10,0)
3	2	$100 x_2^2 + 0.01 \cdot \lvert x_1 + 10 \rvert$	(-10,0)
4	2	$100 \sqrt{\lvert x_2 - 0.01 x_1^2 \rvert} + 0.01 \lvert x_1 + 10 \rvert$	(-10,1)
5	2	$1000 \cdot \lvert x_1^2 + x_2^2 - 800 \rvert + \lvert x_1 + x_2 + 40 \rvert$	(-20,-20)
6	4	$\left[1000 \lvert x_2 - 0.001 x_1^3 \rvert + \lvert x_2 + x_1 + 11 \rvert\right] \times \times \left[1 + 1000 \lvert x_3^2 + x_4^2 - 800 \rvert + \lvert x_3 + x_4 + 40 \rvert\right] + +1000 \lvert x_3^2 + x_4^2 - 800 \rvert + \lvert x_3 + x_4 + 40 \rvert$	(-10,-1,-20,-20)
7	8	$1000 \times \Big\{ (x_1 + 1 - \rho \cos 5\rho)^2 + + (x_2 + 2 - \rho \sin 5\rho \cos 6\rho)^2 + + (x_3 + 3 - \rho \sin 5\rho \sin 6\rho \cos 7\rho)^2 + + (x_4 + 4 - \rho \sin 5\rho \sin 6\rho \sin 7\rho \cos 8\rho)^2 + + (x_5 + 5 - \rho \sin 5\rho \sin 6\rho \sin 7\rho \sin 8\rho \cos 9\rho)^2 + + (x_6 + 6 - \rho \sin 5\rho \sin 6\rho \sin 7\rho \sin 8\rho \sin 9\rho \times \times \cos 10\rho)^2 + + (x_7 + 7 - \rho \sin 5\rho \sin 6\rho \sin 7\rho \sin 8\rho \sin 9\rho \times \times \sin 10\rho \cos 11\rho)^2 + + (x_8 + 8 - \rho \sin 5\rho \sin 6\rho \sin 7\rho \sin 8\rho \sin 9\rho \times \times \sin 10\rho \sin 11\rho)^2 \Big\} + 0.1\rho,$ $\rho = \sqrt{\sum_{i=1}^{8} (x_i + i)^2}.$	(-1,-2,-3,-4,-5,-6,-7,-8)

этой таблице. Все функции имеют единственный минимум, и значение функции в минимуме всегда равно нулю. Начальные значения всех параметров выбраны равными 1. Результаты минимизации этих функций известной программой MINUIT [1] приведены в таблице 3. Использована версия программы 96.03, для более устойчивой минимизации установлено "set strategy 2". В этой же таблице приведены результаты минимизации

Таблица 3: Результат минимизации тестовых функций в виде F_{min}/N_{cal} программой MINUIT (несколько основных режимов) и программой BUKMIN (default mode, т.е. ISTR=1, Inew=1, Isim=2).

№	MINUIT			BUKMIN
	Команда MINUIT	Status	Result	default
1	simplex 1000000	PROGRESS	24.01/515	$\frac{4.4 \cdot 10^{-22}}{54}$
	migrad 1000000	CONVERGED	$4.8 \cdot 10^{-12}/94$	
	minimize 1000000	CONVERGED	$4.8 \cdot 10^{-12}/94$	
	minos 1000000	SUCCESSFUL	$4.8 \cdot 10^{-12}/274$	
2	simplex 1000000	PROGRESS	2.33/ 30	$\frac{1.4 \cdot 10^{-4}}{84}$
	migrad 1000000	CONVERGED	$2.0 \cdot 10^{-8}/87$	
	minimize 1000000	CONVERGED	$2.0 \cdot 10^{-8}/87$	
	minos 1000000	SUCCESSFUL	$2.0 \cdot 10^{-8}/205$	
3	simplex 1000000	PROGRESS	$4.6 \cdot 10^{-4}/33$	$\frac{5.7 \cdot 10^{-6}}{61}$
	migrad 1000000	CONVERGED	$2.6 \cdot 10^{-10}/203$	
	minimize 1000000	CONVERGED	$2.6 \cdot 10^{-10}/203$	
	minos 1000000	PROBLEMS	$2.6 \cdot 10^{-10}/310$	
4	simplex 1000000	PROGRESS	0.22/103	$\frac{2.9 \cdot 10^{-6}}{184796}$
	migrad 1000000	CONVERGED	0.11/166	
	minimize 1000000	CONVERGED	0.11/166	
	minos 1000000	FAILURE	0.11/828	
5	simplex 1000000	PROGRESS	69.30/77	$\frac{5.6 \cdot 10^{-9}}{10910}$
	migrad 1000000	CONVERGED	80.00/121	
	minimize 1000000	CONVERGED	80.00/121	
	minos 1000000	FAILURE	80.00/1305	
6	simplex 1000000	PROGRESS	1650.14/259	$\frac{7.5 \cdot 10^{-6}}{436091}$
	migrad 1000000	FAILED	941.29/461	
	minimize 1000000	FAILED	696.83/1450	
	minos 1000000	FAILURE	751.12/15845	
7	simplex 1000000	PROGRESS	78020.94/1037	$\frac{0.542}{1.3 \cdot 10^{6}}$
	migrad 1000000	CONVERGED	1.74/4322	
	minimize 1000000	CONVERGED	1.74/4322	
	minos 1000000	PROBLEMS	1.74/35806	

этих функций программой BUKMIN в режиме, который устанавливается по умолчанию. По результатам видно, что некоторые функции представляют проблему даже для такой программы, как MINUIT.

В табл. 4 приведены результаты минимизации тестовых функций программой BUKMIN с помощью каждого алгоритма отдельно при двух стратегиях (Istr=0 и Istr=1).

Таблица 4: Результаты минимизации тестовых функций в виде $\frac{F_{min}}{N_{cal}}$ с помощью каждого алгоритма отдельно при двух стратегиях (Istr=0 и Istr=1). Ограничение на число вызовов функции FCN установлено $N_{cal} < 10^6$, требование на точность минимизации $\Delta F_m = 10^{-3}$.

Номер функ.	Результаты для разных алгоритмов					
	VMM	CDM	Simplex	Newton	T&F	Random
Istr = 0						
1	$\frac{2.0\cdot10^{-11}}{110}$	$\frac{4.6\cdot10^{-7}}{199}$	$\frac{12.9}{667}$	$\frac{4.4\cdot10^{-22}}{54}$	$\frac{1.1\cdot10^{6}}{1389}$	$\frac{36312.6}{471}$
2	$\frac{5.1\cdot10^{-6}}{53}$	$\frac{8.0\cdot10^{-5}}{64}$	$\frac{0.00018}{145}$	$\frac{0.00014}{84}$	$\frac{2.128}{52}$	$\frac{0.023}{60}$
3	$\frac{0.0528}{331}$	$\frac{0.110}{29}$	$\frac{8.6\cdot10^{-6}}{79}$	$\frac{5.7\cdot10^{-6}}{61}$	$\frac{0.00015}{48}$	$\frac{0.105}{47}$
4	$\frac{0.1387}{191}$	$\frac{13.94}{10^6}$	$\frac{0.119}{192}$	$\frac{0.100}{84}$	$\frac{0.201}{108}$	$\frac{0.351}{83}$
5	$\frac{77.48}{538}$	$\frac{2.8\cdot10^{5}}{10^6}$	$\frac{78.6}{308}$	$\frac{80.0}{132}$	$\frac{78.46}{115}$	$\frac{77.7}{104}$
6	$\frac{1.4\cdot10^{6}}{382}$	$\frac{1.0\cdot10^{7}}{199}$	$\frac{543.5}{1375}$	$\frac{1.0\cdot10^{7}}{109}$	$\frac{1832.1}{237}$	$\frac{5061.97}{110}$
7	$\frac{0.0404}{1317}$	$\frac{0.130}{793}$	$\frac{300179.1}{1421}$	$\frac{1.665}{193838}$	$\frac{433951.4}{1685}$	$\frac{333.6}{1838}$
Istr = 1						
1	$\frac{1.9\cdot10^{-11}}{130}$	$\frac{4.6\cdot10^{-7}}{199}$	$\frac{0.00011}{9331}$	$\frac{4.4\cdot10^{-22}}{54}$	$\frac{0.0031}{139385}$	$\frac{1004.8}{6729}$
2	$\frac{4.3\cdot10^{-11}}{69}$	$\frac{8.0\cdot10^{-5}}{64}$	$\frac{4.8\cdot10^{-10}}{691}$	$\frac{0.00014}{84}$	$\frac{2.0\cdot10^{-9}}{6990}$	$\frac{9.1\cdot10^{-8}}{723}$
3	$\frac{5.8\cdot10^{-5}}{439}$	$\frac{0.110}{29}$	$\frac{1.8\cdot10^{-7}}{654}$	$\frac{5.7\cdot10^{-6}}{61}$	$\frac{2.8\cdot10^{-5}}{605}$	$\frac{0.0032}{414}$
4	$\frac{0.0435}{10^6}$	$\frac{13.94}{10^6}$	$\frac{0.00072}{152105}$	$\frac{6.7\cdot10^{-5}}{6720}$	$\frac{0.00023}{158965}$	$\frac{0.0328}{15410}$
5	$\frac{67.84}{974}$	$\frac{276100.7}{10^6}$	$\frac{4.2\cdot10^{-7}}{8995}$	$\frac{8.4\cdot10^{-9}}{2410}$	$\frac{2.2\cdot10^{-9}}{11855}$	$\frac{0.00017}{278342}$
6	$\frac{88282.0}{1108}$	$\frac{1.0\cdot10^{7}}{742}$	$\frac{2.8\cdot10^{-5}}{525489}$	$\frac{8.1\cdot10^{6}}{268}$	$\frac{16339.6}{3828}$	$\frac{128.9}{457}$
7	$\frac{0.0320}{3557}$	$\frac{0.130}{793}$	$\frac{0.875}{502437}$	$\frac{1.665}{201130}$	$\frac{1.653}{10^6}$	$\frac{1.132}{17252}$

Результаты получились неоднозначные вследствие очень разных свойств функций. Попробуем рассмотреть их для каждой функции отдельно.

1. Первая функция — простая квадратичная функция, представляющая сложность только из-за очень различающихся собственных значений матрицы квадратичной формы ($\lambda_1 = 1.88$, $\lambda_2 = 124.9$, $\lambda_3 = 12951.3$, $\lambda_4 = 2.08 \cdot 10^6$, $\lambda_5 = 5.44 \cdot 10^8$). MINUIT здесь показала великолепный результат в методе переменной метрики пространства. Методы минимизации в BUKMIN разбиваются на две группы, которые условно можно назвать градиентными (VMM, CDM и Newton) и неградиентными (Simplex, T&F, Random). Первые должны хорошо работать с гладкими, дифференцируемыми функциями и быстро приближаться к минимуму после достижения области, где минимизируемая функция уже хорошо аппроксимируется квадратичной формой. Поэтому закономерно, что и в программе BUKMIN метод переменной метрики пространства, метод сопряжённых направлений и метод Ньютона достигли хорошего результата. Неградиентные методы (Simplex, T&F, Random) в данном случае не могут конкурировать с методами первой группы.

2. Вторая функция тоже является гладкой везде дифференцируемой функцией, но не является квадратичной формой. Поэтому для неё важно, как преобразуются параметры квадратичной формы при приближении к минимуму. В данном случае опять хорошо справились с задачей методы VMM, Ньютона и метод сопряжённых направлений. Симплекс-метод тоже показал неплохой результат.

3. Третья функция везде имеет ограниченный градиент, кроме дна ущелья, где градиент не определён. Поэтому для этой функции самым успешным оказался симплекс-метод. Методы VMM и CDM в программе BUKMIN по мере приближения к минимуму пришли в точку, где не смогли вычислить градиент,— и закончились аварийно. Неожиданно успешным оказался здесь метод Newton.

4. Четвёртая функция отличается от предыдущей тем, что «дно ущелья» представляет собой не прямую линию, как в предыдущей функции, а параболу. Кроме того, при приближении к линии дна ущелья градиент возрастает неограниченно (стремится к бесконечности). Поэтому ни один из алгоритмов программы MINUIT не достиг минимума, а метод переменной метрики пространства

(MIGRAD) даже ошибочно признал конечную точку точкой минимума. В программе BUKMIN наиболее успешным оказался метод Ньютона, но и он не привёл к минимуму.

5. В пятой функции градиент везде ограничен, а на самом дне ущелья не определён. В программе MINUIT ни один из методов не достиг минимума, и метод переменной метрики пространства признал ложную точку минимумом. Правда, команда MINOS, хоть и тоже не достигла минимума, но сообщила о проблемах. В программе BUKMIN самым успешным из градиентных методов оказался метод VMM, а из неградиентных — Random, но ни один из них также не привёл к минимуму.

6. Шестая функция имеет схожие свойства с пятой функцией, но определена в четырёхмерном пространстве. В программе MINUIT ни один из методов не привёл к минимуму, также как и в программе BUKMIN.

7. Седьмая функция интересна тем, что градиент в ней ограничен и определён во всём восьмимерном пространстве, кроме самой точки минимума. Кроме того, путь вдоль ущелья очень извилист и долог. Степень извилистости может характеризовать, например, такой факт, что если начать движение по дну ущелья на расстоянии $\rho = 10$ от минимума (по прямой), то суммарный криволинейный путь будет около 419.5, то есть более, чем в 40 раз превышает прямолинейное расстояние. Функция оказалась настолько трудна, что ни один из алгоритмов ни в программе MINUIT, ни в BUKMIN не достиг цели, но всё же предпочтительнее оказались градиентные методы VMM, CDM и Newton.

Включение стратегии Istr=1 в программе BUKMIN (возможность многократного повторения минимизации из разных начальных точек в случае осознанных проблем) позволяет достичь минимума для первых пяти функций. Минимизация шестой и седьмой функций оказалась непосильной задачей и для MINUIT, и для BUKMIN. Неплохие результаты у симплекс-метода, правда, при худшей эффективности для гладких функций, чем у градиентных методов,— при Istr=1 только седьмую функцию не удалось минимизировать с заданной точностью. Посмотрим, можно ли улучшить результат, используя разные «тандемы» методов минимизации (аналогично команде MINIMIZE в программе Minuit, когда после применения метода MIGRAD в случае неудачи используется метод SIMPLEX). В таблице 5 приведены результаты таких экспериментов.

Таблица 5: Результаты минимизации тестовых функций в виде $\frac{F_{min}}{N_{cal}}$ с помощью разных сочетаний алгоритмов отдельно при двух стратегиях (Istr=1 и Istr=2). Ограничение на число вызовов функции FCN установлено $N_{cal} < 10^7$, требование на точность минимизации $\Delta F_m = 10^{-3}$.

Набор алгор.	Результаты для разных функций						
	1	2	3	4	5	6	7
Istr = 1							
VMM ⇓ Simplex	$\frac{1.9\cdot10^{-11}}{130}$	$\frac{4.3\cdot10^{-11}}{69}$	$\frac{5.8\cdot10^{-5}}{439}$	$\frac{0.0017}{178335}$	$\frac{2.5\cdot10^{-8}}{7612}$	$\frac{0.00025}{161917}$	$\frac{0.032}{3557}$
Newton ⇓ Simplex	$\frac{4.4\cdot10^{-22}}{54}$	$\frac{0.00014}{84}$	$\frac{5.7\cdot10^{-6}}{61}$	$\frac{2.9\cdot10^{-6}}{184796}$	$\frac{5.6\cdot10^{-9}}{10910}$	$\frac{7.5\cdot10^{-6}}{436091}$	$\frac{0.542}{1.3\cdot10^{6}}$
Newton ⇓ VMM	$\frac{4.4\cdot10^{-22}}{54}$	$\frac{0.00014}{84}$	$\frac{5.6\cdot10^{-6}}{61}$	$\frac{6.7\cdot10^{-5}}{12294}$	$\frac{8.4\cdot10^{-9}}{3857}$	$\frac{1111.9}{1313}$	$\frac{1.664}{224708}$
VMM ⇓ Newton	$\frac{1.9\cdot10^{-11}}{130}$	$\frac{4.3\cdot10^{-11}}{69}$	$\frac{5.8\cdot10^{-5}}{439}$	$\frac{0.0037}{1.8\cdot10^{6}}$	$\frac{1.1\cdot10^{-9}}{5886}$	$\frac{350.7}{1545}$	$\frac{0.032}{3557}$
VMM ⇓ Newton ⇓ Simplex	$\frac{1.9\cdot10^{-11}}{130}$	$\frac{4.3\cdot10^{-11}}{69}$	$\frac{5.8\cdot10^{-5}}{439}$	$\frac{3.3\cdot10^{-6}}{90084}$	$\frac{2.0\cdot10^{-10}}{15279}$	$\frac{1.0\cdot10^{-5}}{628075}$	$\frac{0.032}{3557}$
VMM ⇓ Newton ⇓ Simplex ⇓ T&F	$\frac{1.9\cdot10^{-11}}{130}$	$\frac{4.3\cdot10^{-11}}{69}$	$\frac{5.8\cdot10^{-5}}{439}$	$\frac{4.3\cdot10^{-6}}{60841}$	$\frac{5.5\cdot10^{-10}}{19838}$	$\frac{1.4\cdot10^{-5}}{1.54\cdot10^{6}}$	$\frac{0.032}{3557}$
Istr = 2							
VMM ⇓ Simplex	$\frac{2.6\cdot10^{-14}}{6135}$	$\frac{1.1\cdot10^{-13}}{962838}$	$\frac{8.2\cdot10^{-16}}{2848}$	$\frac{0.0017}{186959}$	$\frac{2.5\cdot10^{-8}}{7612}$	$\frac{0.00025}{161917}$	$\frac{2.3\cdot10^{-12}}{39142}$
Newton ⇓ Simplex	$\frac{1.7\cdot10^{-23}}{3752}$	$\frac{2.8\cdot10^{-14}}{856}$	$\frac{2.8\cdot10^{-11}}{655}$	$\frac{2.9\cdot10^{-6}}{184796}$	$\frac{5.6\cdot10^{-9}}{10910}$	$\frac{7.5\cdot10^{-6}}{436091}$	$\frac{0.542}{1.3\cdot10^{6}}$
Newton ⇓ VMM	$\frac{7.7\cdot10^{-23}}{447}$	$\frac{1.5\cdot10^{-22}}{6951}$	$\frac{1.1\cdot10^{-13}}{1850}$	$\frac{6.7\cdot10^{-5}}{12294}$	$\frac{8.4\cdot10^{-9}}{3857}$	$\frac{1111.9}{1313}$	$\frac{1.163}{284659}$
VMM ⇓ Newton	$\frac{1.1\cdot10^{-22}}{725}$	$\frac{9.7\cdot10^{-18}}{380}$	$\frac{9.9\cdot10^{-15}}{1945}$	$\frac{0.0037}{10^{7}}$	$\frac{1.1\cdot10^{-9}}{5886}$	$\frac{350.7}{1545}$	$\frac{7.1\cdot10^{-11}}{26747}$
VMM ⇓ CDM	$\frac{1.5\cdot10^{-17}}{1494}$	$\frac{5.0\cdot10^{-16}}{300}$	$\frac{1.7\cdot10^{-15}}{3319}$	$\frac{0.044}{10^{7}}$	$\frac{2.2\cdot10^{-6}}{114341}$	$\frac{17639.5}{1446}$	$\frac{6.5\cdot10^{-16}}{79371}$
VMM ⇓ Newton ⇓ Simplex	$\frac{5.7\cdot10^{-24}}{4322}$	$\frac{9.7\cdot10^{-18}}{2736}$	$\frac{1.1\cdot10^{-14}}{6222}$	$\frac{3.4\cdot10^{-6}}{90084}$	$\frac{2.0\cdot10^{-10}}{15279}$	$\frac{1.0\cdot10^{-5}}{628075}$	$\frac{3.2\cdot10^{-12}}{1.4\cdot10^{6}}$
VMM ⇓ Newton ⇓ Simplex ⇓ T&F	$\frac{1.8\cdot10^{-23}}{7830}$	$\frac{9.7\cdot10^{-18}}{980}$	$\frac{1.9\cdot10^{-38}}{2381}$	$\frac{4.3\cdot10^{-6}}{60841}$	$\frac{5.5\cdot10^{-10}}{19838}$	$\frac{1.4\cdot10^{-5}}{1.54\cdot10^{6}}$	$\frac{5.5\cdot10^{-13}}{10^{7}}$

Результат, как всегда, неоднозначный. Функции 6 и 7 по-прежнему представляют наибольшую трудность. Если рассматривать только первые 5 функций, то при стратегии Istr=1 хорошо справляются с задачей варианты Newton→Simplex, Newton→VMM, VMM→Newton→Simplex, при этом не затрачивая излишне много времени на более простые функции. Поэтому в качестве стандартной комбинации была выбрана стратегия Istr=1 и тандем алгоритмов Newton→Simplex. Так программа BUKMIN будет минимизировать, если пользователь не поменяет значения соответствующих переменных в общем блоке /BUKMIN00/. Такой выбор можно охарактеризовать коротко — «не хуже MINUIT, быстро и достаточно точно для функций простых и средней сложности» (на основании данных о результате минимизации в таблицах).

При Istr=2 надёжнее всего минимизирует «связка» программ VMM→Newton→Simplex или VMM→Newton→Simplex→T&F — минимизирует все функции, правда, при этом на минимизацию более простых функций тратит неоправданно много процессорного времени.

Таблица 6 демонстрирует результаты минимизации этих же семи функций при стратегии Istr=3.

При этой стратегии тоже есть успешная последовательность программ, которые сумели найти минимумы всех функций — VMM→Newton→Simplex→T&F.

Для полноты картины осталось проверить, как справляется программа BUKMIN с минимизацией функций при очень большом числе параметров (возьмём для проверки $n = 100$). Выберем для минимизации следующие функции:

8. Квадратичная форма

$$F_8 = \sum_{k=1}^{n-1} \left[\sum_{j=1}^{k} (x_j + j) - k \cdot (x_{k+1} + k + 1) \right]^2 + \left[\sum_{j=1}^{n} (x_j + j) \right]^2 .$$

   Одно собственное значение равно n, остальные выражаются формулой $\lambda_k = k \cdot (k+1)\,,\ \ k = 1, 2, \ldots, n-1$.

9. Гладкая функция с извилистым ущельем

$$F_9 = (x_1 + 10)^2 + \left[1000 + (x_1 + 10)^2 \right] \cdot \left[\sum_{k=2}^{n} \left(x_k + 10 \sin \frac{k \cdot x_1}{k-1} \right)^2 \right] .$$

10. Функция с извилистым ущельем, у которой градиент везде существует и ограничен, но не определён на дне ущелья

$$F_{10} = (x_1 + 10)^2 + \left[1000 + (x_1 + 10)^2\right] \cdot \sqrt{\sum_{k=2}^{n} \left(x_k + 10 \sin \frac{k \cdot x_1}{k-1}\right)^2}.$$

11. Функция с извилистым ущельем, у которой градиент неограниченно возрастает при приближении ко дну ущелья

$$F_{11} = (x_1 + 10)^2 + \left[1000 + (x_1 + 10)^2\right] \cdot \sqrt[4]{\sum_{k=2}^{n} \left(x_k + 10 \sin \frac{k \cdot x_1}{k-1}\right)^2}.$$

Таблица 6: Результаты минимизации тестовых функций в виде $\frac{F_{min}}{N_{cal}}$ с помощью разных сочетаний алгоритмов при стратегии Istr=3. Ограничение на число вызовов функции FCN установлено $N_{cal} < 10^7$, требование на точность минимизации $\Delta F_m = 10^{-3}$.

Набор алгор.	Результаты для разных функций						
	1	2	3	4	5	6	7
VMM	$\frac{7.1 \cdot 10^{-25}}{1393}$	$\frac{1.2 \cdot 10^{-25}}{1470}$	$\frac{5.3 \cdot 10^{-16}}{7445}$	$\frac{0.057}{9726}$	$\frac{77.48}{9481}$	$\frac{310.0}{12944}$	$\frac{9.2 \cdot 10^{-13}}{43552}$
CDM	$\frac{2.5 \cdot 10^{-18}}{13194}$	$\frac{2.3 \cdot 10^{-16}}{1216}$	$\frac{1.1 \cdot 10^{-16}}{3872}$	$\frac{0.112}{2.1 \cdot 10^6}$	$\frac{0.00069}{4.2 \cdot 10^6}$	$\frac{574.3}{10886}$	$\frac{1.9 \cdot 10^{-13}}{22417}$
Simplex	$\frac{3.2 \cdot 10^{-13}}{344810}$	$\frac{5.0 \cdot 10^{-23}}{5030}$	$\frac{3.8 \cdot 10^{-14}}{2807}$	$\frac{2.4 \cdot 10^{-7}}{105314}$	$\frac{1.6 \cdot 10^{-5}}{39868}$	$\frac{0.400}{108266}$	$\frac{1.790}{10^7}$
Newton	$\frac{7.7 \cdot 10^{-23}}{107}$	$\frac{6.5 \cdot 10^{-22}}{3531}$	$\frac{8.3 \cdot 10^{-13}}{749}$	$\frac{0.100}{4239}$	$\frac{1.3 \cdot 10^{-9}}{7600}$	$\frac{79.98}{11794}$	$\frac{1.660}{1.8 \cdot 10^6}$
T&F	$\frac{2.7 \cdot 10^{-5}}{968129}$	$\frac{8.6 \cdot 10^{-14}}{38374}$	$\frac{2.9 \cdot 10^{-11}}{6764}$	$\frac{0.00020}{15207}$	$\frac{3.0 \cdot 10^{-6}}{20401}$	$\frac{80.0}{154977}$	$\frac{1.655}{841178}$
Random	$\frac{21.92}{439501}$	$\frac{1.3 \cdot 10^{-11}}{10296}$	$\frac{1.2 \cdot 10^{-9}}{3953}$	$\frac{0.081}{3342}$	$\frac{0.031}{5583}$	$\frac{101.17}{4843}$	$\frac{1.493}{62686}$
VMM ⇓ Simplex	$\frac{1.2 \cdot 10^{-26}}{10158}$	$\frac{2.1 \cdot 10^{-31}}{5446}$	$\frac{3.6 \cdot 10^{-14}}{2949}$	$\frac{6.8 \cdot 10^{-7}}{126535}$	$\frac{2.4 \cdot 10^{-6}}{41565}$	$\frac{0.017}{185234}$	$\frac{2.5 \cdot 10^{-12}}{105585}$
CDM ⇓ Simplex	$\frac{1.0 \cdot 10^{-19}}{40981}$	$\frac{3.9 \cdot 10^{-24}}{3903}$	$\frac{7.3 \cdot 10^{-16}}{2118}$	$\frac{1.3 \cdot 10^{-6}}{2.5 \cdot 10^6}$	$\frac{1.4 \cdot 10^{-7}}{4.2 \cdot 10^6}$	$\frac{0.027}{159103}$	$\frac{2.6 \cdot 10^{-15}}{24893}$
Newton ⇓ Simplex	$\frac{2.0 \cdot 10^{-22}}{1851}$	$\frac{7.5 \cdot 10^{-25}}{4905}$	$\frac{3.0 \cdot 10^{-13}}{4383}$	$\frac{4.1 \cdot 10^{-7}}{136342}$	$\frac{2.9 \cdot 10^{-9}}{31400}$	$\frac{0.00043}{137040}$	$\frac{1.485}{10^7}$
Newton ⇓ T&F	$\frac{7.7 \cdot 10^{-23}}{1947}$	$\frac{2.6 \cdot 10^{-21}}{6548}$	$\frac{8.3 \cdot 10^{-13}}{3572}$	$\frac{6.4 \cdot 10^{-5}}{20995}$	$\frac{9.2 \cdot 10^{-12}}{22606}$	$\frac{79.99}{99161}$	$\frac{1.659}{2.5 \cdot 10^6}$
VMM ⇓ Newton ⇓ Simplex	$\frac{8.8 \cdot 10^{-23}}{2102}$	$\frac{3.4 \cdot 10^{-24}}{3183}$	$\frac{5.9 \cdot 10^{-14}}{2043}$	$\frac{2.6 \cdot 10^{-7}}{133364}$	$\frac{2.4 \cdot 10^{-12}}{50278}$	$\frac{31.88}{91292}$	$\frac{3.0 \cdot 10^{-13}}{124153}$
VMM ⇓ Newton ⇓ Simplex ⇓ T&F	$\frac{8.8 \cdot 10^{-23}}{3906}$	$\frac{2.0 \cdot 10^{-27}}{2790}$	$\frac{8.3 \cdot 10^{-37}}{5430}$	$\frac{1.8 \cdot 10^{-6}}{134265}$	$\frac{3.1 \cdot 10^{-9}}{60241}$	$\frac{4.7 \cdot 10^{-5}}{316392}$	$\frac{1.2 \cdot 10^{-12}}{294238}$

Обычно программа MINUIT устанавливается с ограничением $n \leq n_{max} = 50$ на число переменных параметров функции. Для того, чтобы проверить успешность работы программы MINUIT на этих функциях, была специально проведена установка программы MINUIT с бóльшим ограничением $n_{max} = 200$ (тот же самый Release 96.03). В табл. 7 приведены результаты минимизации этих четырёх функций программой MINUIT, а также программой BUKMIN в режиме, устанавливаемом по умолчанию. Как и ожидалось, первые две функции оказались вполне по силам методу переменной метрики пространства.

Таблица 7: Результат минимизации тестовых функций с $n = 100$ в виде F_{min}/N_{cal} программой MINUIT (несколько основных режимов) и программой BUKMIN ($N_{cal} < 10^7$, default mode, т.е. ISTR=1, Inew=1, Isim=2).

№	MINUIT			BUKMIN
	Команда MINUIT	Status	Result	default
8	simplex 1000000	PROGRESS	$9.1 \cdot 10^7/251284$	$\frac{5.5 \cdot 10^{-25}}{309262}$
	migrad 1000000	CONVERGED	$9.1 \cdot 10^{-11}/11521$	
	minimize 1000000	CONVERGED	$9.1 \cdot 10^{-12}/11521$	
	minos 1000000	PROBLEMS	$9.1 \cdot 10^{-12}/97905$	
9	simplex 1000000	PROGRESS	1256.8/113975	$\frac{9.4 \cdot 10^{-7}}{6098783}$
	migrad 1000000	CONVERGED	$8.9 \cdot 10^{-5}/310005$	
	minimize 1000000	CONVERGED	$8.9 \cdot 10^{-5}/310005$	
	minos 10000000	PROBLEMS	$8.9 \cdot 10^{-5}/2361700$	
10	simplex 1000000	PROGRESS	880.6/85543	$\frac{98.0}{10^7}$
	migrad 1000000	FAILED	71.5/28973	
	minimize 1000000	FAILED	71.5/49594	
	minos 10000000	PROBLEMS	71.5/4940543	
11	simplex 1000000	PROGRESS	1034.5/75089	$\frac{573.2}{10^7}$
	migrad 1000000	CONVERGED	79.8/31167	
	minimize 1000000	CONVERGED	73.0/63566	
	minos 10000000	FAILURE	76.4/2690463	

В таблице 8 приведены результаты минимизации этих функций программой BUKMIN в разных режимах.

Таблица 8: Результаты минимизации тестовых функций с большим числом параметров $n = 100$ в виде F_{min}/N_{cal} с помощью разных сочетаний алгоритмов и разных стратегиях Istr. Ограничение на число вызовов функции FCN установлено $N_{cal} < 10^7$, требование на точность минимизации $\Delta F_m = 10^{-3}$.

Набор алгор.	Результаты для разных функций			
	8	9	10	11
Istr=0				
VMM	$6.9 \cdot 10^{-9}/20248$	0.880/11572	0.838/12329	34.98/11123
CDM	0.0028/234680	0.827/87145	0.763/22881	39.2/44187
Simplex	$3.7 \cdot 10^{7}/36334$	2951.1/39311	1431.3/21618	1130.8/19440
Newton	$2.0 \cdot 10^{-24}/309262$	94.2/41342	$98.19/1.8 \cdot 10^{6}$	$226.7/9.6 \cdot 10^{6}$
T&F	247649.0/34196	97.0/12428	97.0/14832	97.0/18450
Randon	3750.1/311929	431.6/17873	753.1.0/17838	592.2/17781
Istr=1				
VMM	$6.9 \cdot 10^{-9}/20644$	$1.4 \cdot 10^{-12}/164478$	$0.099/5.0 \cdot 10^{6}$	$2.42/2.3 \cdot 10^{6}$
CDM	0.0028/234680	0.827/87145	0.635/53569	$7.46/6.7 \cdot 10^{6}$
Simplex	$2.3 \cdot 10^{7}/2.2 \cdot 10^{6}$	$88.5/3.4 \cdot 10^{6}$	$94.2/1.7 \cdot 10^{6}$	$7.73/9.7 \cdot 10^{6}$
Newton	$2.0 \cdot 10^{-24}/309262$	$1.66/7.4 \cdot 10^{6}$	$97.66/10^{7}$	$226.7/10^{7}$
VMM ⇓ Newton	$6.8 \cdot 10^{-9}/20644$	$1.4 \cdot 10^{-12}/164478$	$0.0017/8.9 \cdot 10^{6}$	$1.12/10^{7}$
VMM ⇓ Newton ⇓ Simplex	$6.9 \cdot 10^{-9}/20644$	$1.4 \cdot 10^{-12}/164478$	$5.5 \cdot 10^{-5}/5.8 \cdot 10^{6}$	$0.0040/10^{7}$
Istr=2				
VMM	$4.0 \cdot 10^{-17}/676356$	$1.2 \cdot 10^{-13}/1.6 \cdot 10^{6}$	$0.099/5.0 \cdot 10^{6}$	$2.42/2.3 \cdot 10^{6}$
CDM	$5.1 \cdot 10^{-23}/546438$	$7.1 \cdot 10^{-16}/1.5 \cdot 10^{6}$	$0.636/53569$	$7.46/6.7 \cdot 10^{6}$
Newton	$2.0 \cdot 10^{-24}/639329$	$1.4 \cdot 10^{-6}/8.2 \cdot 10^{6}$	$97.66/10^{7}$	$226.7/10^{7}$
VMM ⇓ Newton	$2.8 \cdot 10^{-25}/88094$	$1.3 \cdot 10^{-12}/896821$	$0.0017/8.9 \cdot 10^{6}$	$1.12/10^{7}$
VMM ⇓ Newton ⇓ Simplex	$2.8 \cdot 10^{-25}/1.8 \cdot 10^{6}$	$9.2 \cdot 10^{-13}/1.9 \cdot 10^{6}$	$5.5 \cdot 10^{-5}/5.8 \cdot 10^{6}$	$0.0040/10^{7}$
Istr=3				
VMM	$2.2 \cdot 10^{-20}/420519$	$5.0 \cdot 10^{-17}/4.0 \cdot 10^{6}$	$0.822/747115$	$1.28/1.1 \cdot 10^{6}$
CDM	$5.7 \cdot 10^{-20}/365985$	$6.5 \cdot 10^{-18}/4.5 \cdot 10^{6}$	$0.682/370871$	$7.46/354155$
Newton	$2.0 \cdot 10^{-24}/324716$	$94.23/5.5 \cdot 10^{6}$	$98.19/10^{7}$	$226.7/10^{7}$
VMM ⇓ Newton	$3.0 \cdot 10^{-28}/43304$	$1.4 \cdot 10^{-18}/2.1 \cdot 10^{6}$	$0.0054/4.0 \cdot 10^{6}$	$1.33/10^{7}$
VMM ⇓ Newton ⇓ Simplex	$3.0 \cdot 10^{-28}/882205$	$3.1 \cdot 10^{-18}/4.6 \cdot 10^{6}$	$0.0060/10^{7}$	$0.0042/10^{7}$

Для пространства большой размерности метод Ньютона оказался не такой успешный, как для пространства малой и средней размерности. Это объясняется тем, что вычисление матрицы квадратичной формы вдали от минимума при большой размерности страдает от накопления ошибок округления, кроме того, для вычисления матрицы и совершения очередного шага в сторону минимума требуется вычислить гораздо больше опорных значений функции, чем в методе переменной метрики пространства. Для пространства большой размерности тоже довольно успешны оказываются сочетания алгоритмов программы VMM→Newton и VMM→Newton→Simplex.

12 Заключение

Разработана программа BUKMIN безусловной численной минимизации функции многих переменных. Предполагается её использование при обработке экспериментальных данных на детекторах СНД, КМД, КЕДР. Для достаточно простых функций качество минимизации этой программы сравнимо с качеством широко известной программы MINUIT. Сравнительные достоинства и недостатки этой программы можно кратко сформулировать следующим образом.

Достоинства:

- Программа написана на языке Фортран и не использует никаких специальных библиотек. Несмотря на большое количество алгоритмов минимизации в программе BUKMIN (переменной метрики пространства, метод сопряжённых направлений, метод Ньютона, симплекс-метод, метод проб и ошибок, метод случайного поиска), которые можно использовать в любых сочетаниях, программа достаточно компактна — около 3000 строк исходного кода. Это означает, что необходимым и достаточным условием использования данной программы является наличие компилятора Fortran-77.
- Простая модульная организация программы позволяет легко добавлять новые алгоритмы минимизации.
- Некоторые сложные функции программа BUKMIN минимизирует более надёжно, чем MINUIT. Это даёт возможность даже в тех случаях, когда MINUIT доступна для использования, проверить качество минимизации альтернативной программой.

Недостатки:

- Любая новая программа, не отличающаяся кардинально от ранее написанных, имеет естественный недостаток — малый опыт эксплуатации, ненулевая вероятность присутствия какой-либо ошибки. Правда, этот недостаток со временем исчезает.
- Следующий недостаток является продолжением достоинства — компактность и простота программы обусловливает отсутствие многих приятных функций программы MINUIT. Это отсутствие процедуры оценки статистических ошибок для случая минимизации логарифмической функции правдоподобия, невозможность нарисовать карту линий уровня на двухкоординатном срезе профиля функции, невозможность ограничить область изменения параметров и т.п. Часть этих недостатков можно компенсировать, построив свою аналогичную процедуру, используя для минимизации программу BUKMIN, но в любом случае это дополнительная работа.
- Отсутствие идентичной версии программы на популярном в настоящее время языке C++ существенно ограничивает возможность её использования.

Этот препринт можно рассматривать как подробное описание программы.

Список литературы

[1] *F.James, M.Roos.* "MINUIT" a system for function minimization and analysis of the parameter errors and correlations. Computer Physics Communications, 10 (1975) 343-367.

[2] *M.N.Achasov, V.M.Aulchenko, S.E.Baru et al.* Spherical Neutral Detector for VEPP-2M collider. Nucl. Instr. and Meth., A449 (2000) 125-139. e-Print: hep-ex/9909015.

[3] *R.Fletcher, M.J.D.Powell.* A rapidly converging descent method for minimization. Comput. J., 6 (1963) 163.

[4] *А.Д.Букин, С.И.Эйдельман.* ЭВМ в планировании и обработке эксперимента. Учеб. пособие. Изд. 2-е. Новосиб. гос. ун-т, Новосибирск, 2002.

[5] *A.I.Manevich, E.Boudinov.* An efficient conjugate directions method without linear minimization. Nucl. Instr. and Meth. in Phys. Research, A 455 (2000) 698-705.

[6] *J.A.Nelder, R.Mead.* A simplex method for function minimization. Comput. J., 7 (1965) 308.

Содержание

А.Д. Букин

Программа численной минимизации функции многих параметров

A.D. Bukin

Subroutine for numerical minimization of the function of many parameters

ИЯФ 2004-78

Ответственный за выпуск А.М. Кудрявцев
Работа поступила 17.12.2004 г.

Сдано в набор 20.12.2004 г.
Подписано в печать 20.12.2004 г.
Формат бумаги 60×90 1/16 Объем 2.4 печ.л., 1.9 уч.-изд.л.
Тираж 100 экз. Бесплатно. Заказ № 78

Обработано на IBM PC и отпечатано на
ротапринте ИЯФ им. Г.И. Будкера СО РАН
Новосибирск, 630090, пр. академика Лаврентьева, 11.