Программа минимизации функции многих переменных методом деформируемого многогранника (по Нелдеру и Миду)

Цибанов В.В.
tsibanoff@gmail.com

Реализован алгоритм минимизации функции многих переменных без вычисления производных. В основу положен метод деформируемого многогранника, известный также как модифицированный симплекс-метод. Программа ("Simplex") написана на языке Excel Vusual Basic и опубликована в сети Интернет для свободного пользования. Предлагается в качестве простого и эффективного средства решения широкого круга вычислительных задач, таких как: решение линейных и нелинейных уравнений и их систем, минимизация суммы квадратов для линейных и нелинейных регрессий, поиск экстремумов функций без ограничений и с ограничениями в виде равенств и неравенств и т.п. Описан алгоритм и порядок использования программы, дан ряд примеров решения наиболее типичных задач.

Алгоритм деформируемого многогранника (модифицированный симплекс-метод) для минимизации функции многих переменных был предложен Нелдером и Мидом в 1964 г. [1]. Относится к категории методов оптимизации, не использующих производные. Показав свою эффективность при относительной простоте, стал своего рода "стандартом" в категории нелинейного программирования, вошел во многие учебники, монографии и курсы по прикладным математическим методам [2]. Реализовывался на ЭВМ неоднократно. В настоящей работе он представлен в виде программы-макроса "Simplex" на внутреннем языке VBA (Visual Basic for Applications) популярного приложения MS Office Excel (для ПК, оснащённых ОС Windows). Наибольшее внимание уделено рассмотрению конкретных примеров использования программы.

Постановка задачи и суть рассматриваемого метода в упрощенном изложении сводится к следующему.

Имеется целевая функция (ЦФ): $F = F(\vec{p})$, где $\vec{p} = (p_1, p_2, ..., p_m)^T$ – вектор (набор) её параметров (T – знак транспонирования – замены строки на столбец; вектор является столбцом). Требуется найти такое значение вектора, $\vec{p}^* = (p_1^*, p_2^*, ..., p_m^*)^T$, при котором F принимает минимальное значение: $F^* = F(\vec{p}^*) = \min$. Для этого в пространстве параметров и целевой функции размерности $m + 1$ из некоторой начальной точки $\vec{p}^{(0)} = (p_1^{(0)}, p_2^{(0)}, ..., p_m^{(0)})^T$ строится по определённым правилам равносторонний (изначально) многогранник, называемый симплексом, длина рёбер которого определяется шагом симплекса s. В $m + 1$ вершинах симплекса вычисляются значения F. Далее эти значения сравниваются и выбираются 2 вершины – лучшая и худшая – где значения F, соответственно, наименьшее и наибольшее. Для оценки направления, в котором функция уменьшается, производится *отражение* худшей вершины относительно центроида многогранника. Если в отражённой точке значение F получилось ещё меньше, производится операция *растяжения* симплекса. Всякий раз при удачном шаге координаты худшей точки заменяется координатами вновь найденной лучшей. При неудачных шагах, т.е. когда не удаётся найти точки, где F меньше, чем уже было достигнуто, предусмотрен поиск второго большего значения, а также *сжатие* симплекса по текущему направлению и его *редукция* – уменьшение всех сторон многогранника. В целом, по мере продвижения к минимуму F, многогранник изменяет свою конфигурацию, приспосабливаясь к топологии гиперповерхности ЦФ, а в окрестностях самого минимума уменьшает свой объём до некоторой заданной предельной величины ε. Когда это происходит, считается, что минимум F найден. Более детально алгоритм метода и программы представлен в виде блок-схемы на рис. 1.

Идентификаторы программы.

а) целые (integer):

m – размерность вектора (число) параметров ЦФ;

n – число точек регрессии (когда программа используется как метод наименьших квадратов, МНК);

i1 – количество вычислений ЦФ;

L – номер вершины многогранника, в которой значение ЦФ минимально; H – то же, где оно максимально;

i, j, k, u – индексы; k1, k2, k3, k4 – вспомогательные;

б) константы:

alfa – коэффициент отражения; beta – коэффициент сжатия; gamma – коэффициент растяжения симплекса; step – величина шага; eps – минимальный объём многогранника; d1, d2 – вспомогательные;

в) переменные:

F2 – целевая функция, ЦФ; F1 – функция-модель (используется в случае МНК);

S1 – второе наибольшее значение ЦФ в многограннике; C1 – вспомогательная;

mnk – логическая переменная (Boolean), имеющая значения: True. когда программа используется как МНК, иначе – False;

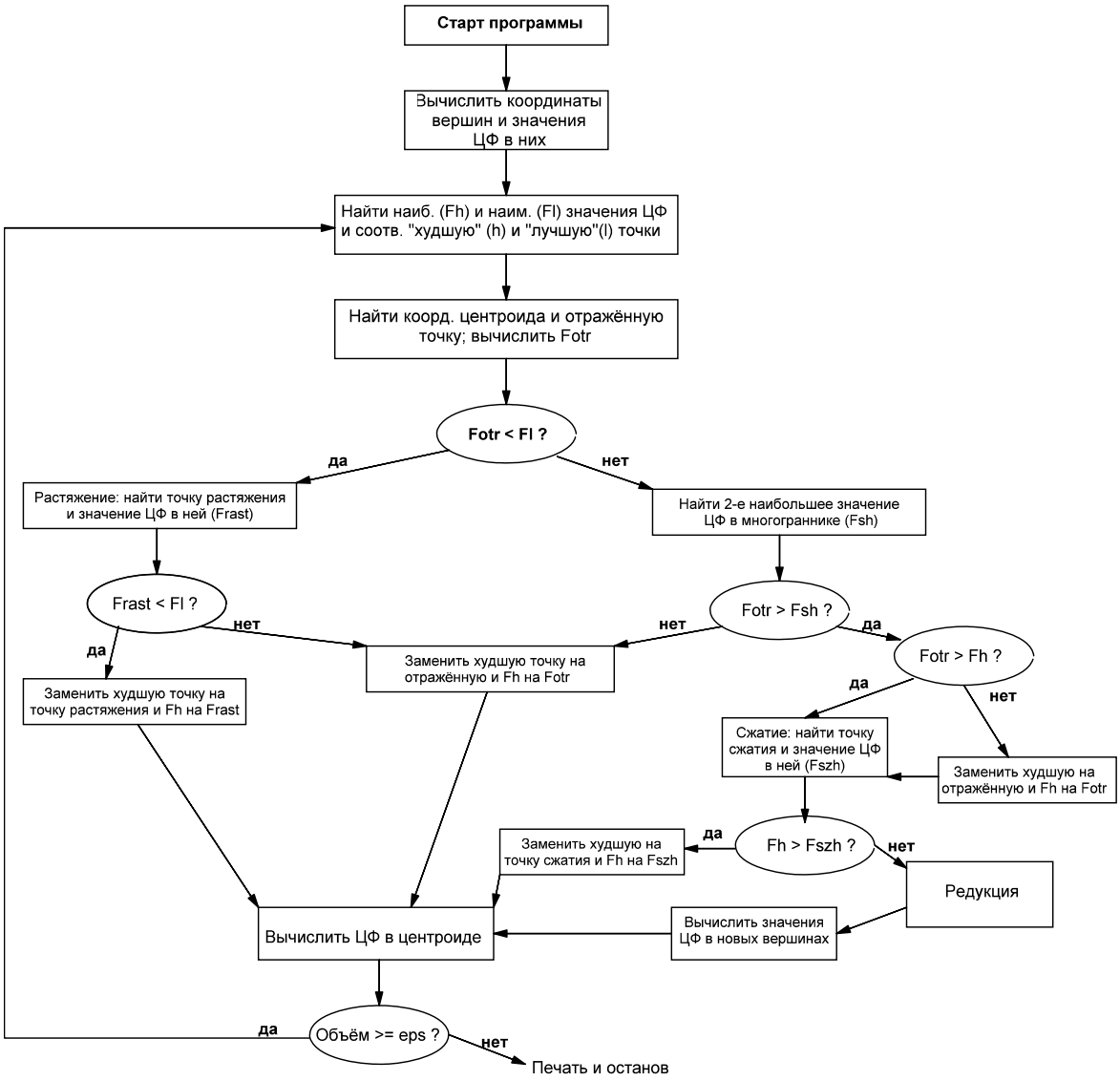

Рис. 1. Блок-схема алгоритма и программы "Simplex".

г) массивы:

P(1 to m) – вектор параметров ЦФ;

X(1 to n) – массив независимой переменной (абсцисса регрессии) в МНК;

Y(1 to n) – массив наблюдений (ордината регрессии) в МНК;

S(1 to m + 4, 1 to m) – массив координат вершин и точек многогранника;

F(1 to m + 4) – массив значений ЦФ в вершинах и точках многогранника.

Структура массива величин S(i , j) следующая:

индексы принимают значения: i = 1, …, m + 4; j = 1, …, m.

S(1, j) – координаты исходной точки $\vec{p}^{(0)} = \left(p_1^{(0)}, p_2^{(0)}, ..., p_m^{(0)}\right)^T$;

от S(2 ,j) до S(m +1, j) – координаты вершин симплекса;

S(m + 2, j) – координаты центроида;

S(m + 3, j) – отражённая точка;

S(m + 4, j) – точка растяжения или сжатия.

Порядок пользования программой.

Программа написана на языке Basic в среде VBA (Visual Basic for Applications). Это встроенный язык программирования Excel в пакете MS Office [3]. Для использования программы, необходимо, чтобы на ПК, оснащённом ОС MS Windows XP или более поздней, было установлено приложение MS Office Excel[1].

Проделать следующее:

1. Скачать и *сохранить* файл программы simplex-v1.xls, пройдя по ссылке [4].
2. Открыть приложение Excel, не открывая скачанного файла.
3. Т.к. программа является *макросом*, необходимо снизить уровень безопасности приложения, для чего войти в основном меню по пути: Сервис > Макрос > Безопасность; на открывшейся вкладке выбрать *средний* уровень безопасности.
4. Открыть файл программы и в открывшемся окне предупреждения системы безопасности выбрать «Не отключать макросы». Появится главное окно программы – "Simplex".

Кроме того, имеются ещё две вкладки: "Константы" и "МНК". Из констант следует ввести во второй столбец: "число параметров минимизируемой функции, m" и "используется ли Simplex как МНК (True или False)". В случае МНК нужно ввести также число точек регрессии, *n*. Прочие константы установлены по умолчанию и их можно не менять. На вкладке "МНК" размещается регрессионная таблица, где в соответствующем случае заполняются столбцы X(u) и Y(u), согласно заданной величине *n* (1-ый столбец можно не заполнять).

Для просмотра программного кода и занесения необходимых описаний конкретных функций надо пройти из основного меню по пути: Сервис > Макрос > Редактор Visual Basic либо нажать <Alt+F11>. Откроется окно Microsoft Visual Basic – simplex-v1.xls с текстом программы[2].

Примечание: в Excel 2010 для запуска редактора Visual Basic необходимо перейти на вкладку "Разработчик" и в группе Код нажать кнопку Visual Basic: откроется интегрированная среда разработки приложений Visual Basic. Если на *ленте* в Microsoft Office Excel 2010 нет вкладки "Разработчик", необходимо выполнить следующие действия: 1. Перейти на вкладку Файл и нажать на кнопку Параметры. 2. В открывшемся окне Параметры Excel выбрать слева категорию "Настройка ленты", а справа в группе "Настройка ленты" в раскрывающемся списке "Основные вкладки" установить флажок "Разработчик" и нажать на кнопку ОК. Для быстрого запуска редактора VBA достаточно нажать комбинацию клавиш <Alt+F11>.

---

[1] Следующее относится к Excel 2003.

[2] Если данное окно пустое, то из его главного меню пройти: View > Project Explorer (или нажать <Ctrl + r>) и в открывшемся окошке Project-VBAProject дважды щёлкнуть на Module1.

Программа составлена из следующих процедур (Subroutines):

1. Sub Simplex() − собственно программа симлекс-метода по Нелдеру и Миду.
2. Sub OF() − процедура вычисления ЦФ, F2 (Objective Function).
3. Sub MF(u) – процедура вычисления функции-модели F1 в МНК (Model Function), где u – зарезервированный индекс (public integer), относящийся к u-ой точке регрессии. Важное замечание: индексы i, j также являются зарезервированными (для массивов S и F), и применять их в процедурах для функций не следует.
4. Sub RS() – процедура *восстановления* симплекса (Restore Simplex); применяется в случае непредвиденного коллапса многогранника; о ней будет сказано в конце статьи.

Дальнейший порядок работы с программой лучше всего рассмотреть на конкретных примерах.

1⁰. Минимизация функции без ограничений; m = 2.

Найти минимум функции 2-х переменных:

$$F(\vec{p}) = 100 \cdot \left(p_2 - p_1^{\,2}\right)^2 + (1 - p_1)^2 .$$

Это т.н. *функция Розенброка*, отличающаяся тем, что в пространстве параметров в области своего минимума $\vec{p}^{\,*} = (1, 1)^T$ представляет собой узкий искривлённый овраг. Задача минимизации овражистых функций особенно затруднительна. Поэтому функцию Розенброка обычно используют как тестовую задачу в оценках эффективности тех или иных методов минимизации. Записываем процедуру вычисления ЦФ в виде[3]:

```
Sub OF() 'функция Розенброка
  F2 = 100 * (P(2) - P(1) ^ 2) ^ 2 + (1 - P(1)) ^ 2
  i1 = i1 + 1
End Sub
```

Примечание: последний оператор служит для подсчёта количества обращений к процедуре.

На вкладке "Константы" в соответствующие ячейки вводим: m → 2; mnk → False.

На вкладке "Simplex" вводим начальное приближение (в данном примере оно м.б. любым), например, нули.

Запускаем программу нажатием <Ctrl+s>. При заданной величине eps = 10^{-8} получаем результат: $\vec{p}^{\,*} = (0.9998423,\ 0.9996637)^T$; $F(\vec{p}^{\,*}) = 6.852_{10}\text{-}08$ (i1 = 99). При eps = 10^{-10}: $\vec{p}^{\,*} = (0.9999859,\ \ 0.9999709)^T$; $F(\vec{p}^{\,*}) = 2.580_{10}\text{-}10$ (i1 = 115).

2⁰. Минимизация функции без ограничений; m = 4.

Еще одной функцией, которая считается сложной с точки зрения поиска экстремума и используется для испытаний численных методов оптимизации, является функция Вуда:

$$F(\vec{p}) = 100\left(p_1^2 - p_2\right)^2 + (1 - p_1)^2 + 90\left(p_3^2 - p_4\right)^2 + (1 - p_3)^2 + 10.1\left((1 - p_2)^2 + (1 - p_4)^2\right) + 19.8(1 - p_2)(1 - p_4)$$

Ее минимум $F(\vec{p}^{\,*}) = 0$ расположен в точке $\vec{p}^{\,*} = 1$. Процедура для ЦФ будет выглядеть так:

```
Sub OF() 'функция Вуда
F2 = 100 * (P(1) ^ 2 - P(2)) ^ 2 + (1 - P(1)) ^ 2 + 90 * (P(3) ^ 2 - P(4)) ^ 2 + (1 - P(3)) ^ 2 _
+ 10.1 * ((1 - P(2)) ^ 2 + (1 - P(4)) ^ 2) + 19.8 * (1 - P(2)) * (1 - P(4))
i1 = i1 + 1
End Sub
```

На вкладке "Константы" в соответствующие ячейки вводим: m → 4; mnk → False.

---

[3] В программе по умолчанию Sub OF( ) записана в более общем виде.

На вкладке "Simplex" вводим начальное приближение. В данном примере оно также м.б. любым, но одним из "худших" почему-то считается $\vec{p}^{(0)} = (-3;-1;-3;-1)^T$. Введем его и при eps = 10^{-7} получим (параметры округлены до десятитысячных): $F(\vec{p}^*)$ = 4.728_{10}-7; $\vec{p}^* = (1.0002;1.0005;0.9997;0.9994)^T$; i1 = 235.

3^0. Решение системы 2-х нелинейных уравнений.

Рассмотрим практическую задачу из области радиотехники. Необходимо определить ёмкости растягивающих конденсаторов $C1$ и $C2$ колебательного контура (рис. 2), такие, чтобы при перестройке конденсатора переменной ёмкости от $Ct_{\min}$ до $Ct_{\max}$ ёмкость триады менялась в пределах от $C0_{\min}$ до $C0_{\max}$.

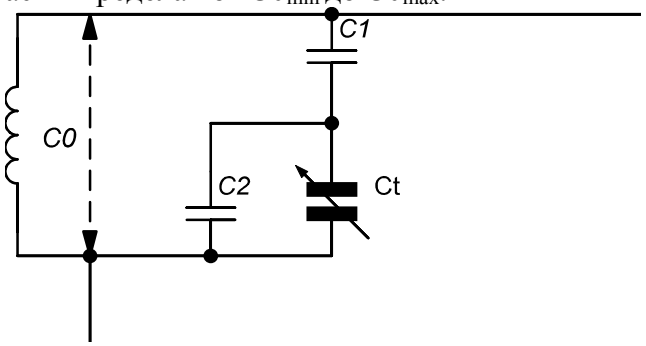

Рис. 2. К расчёту конденсаторов растяжки.

Имеем систему двух нелинейных уравнений с двумя неизвестными C1 и C2:

$$\left.\begin{aligned} C0_{\min} &= C1\cdot(C2+Ct_{\min})/(C1+C2+Ct_{\min}) \\ C0_{\max} &= C1\cdot(C2+Ct_{\max})/(C1+C2+Ct_{\max}) \end{aligned}\right\}.$$

Вместо того, чтобы решать ее алгебраически, путем исключения одного неизвестного и сведения к квадратному уравнению для другого неизвестного, при неизбежных громоздких выкладках, применим программу "Simplex".

Перепишем систему так:

$$\left.\begin{aligned} f_1 &\equiv C0_{\min} - C1\cdot(C2+Ct_{\min})/(C1+C2+Ct_{\min}) = 0 \\ f_2 &\equiv C0_{\max} - C1\cdot(C2+Ct_{\max})/(C1+C2+Ct_{\max}) = 0 \end{aligned}\right\}$$

и зададим ЦФ в виде:

$$F = f_1^2 + f_1^2 .$$

Ясно, что F равна нулю тогда и только тогда, когда оба слагаемых равны нулю, т.е. совокупность параметров $C1$ и $C2$, при которых значение F минимально, и будет решением задачи. Пусть конкретно (в пФ): $Ct_{\min}$ = 10; $Ct_{\max}$ = 500; $C0_{\min}$ = 60; $C0_{\max}$ = 200. Обозначаем $C1\rightarrow$P(1); $C2\rightarrow$P(2). Процедура вычисления ЦФ может выглядеть так:

```
Sub OF()
F2 = (60 - P(1) * (P(2) + 10) / (P(1) + P(2) + 10)) ^ 2
F2 = F2 + (200 - P(1) * (P(2) + 500) / (P(1) + P(2) + 500)) ^ 2
i1 = i1 + 1
End Sub
```

Запускаем программу и получаем (при eps = 10^{-7} и старте из начала координат, округлённо): $C1$ = 310 пФ; $C2$ = 64 пФ; значение ЦФ в минимуме = 7.787_{10}-8 ; i1 = 115.

4^0. Использование программы в качестве МНК. Линейная регрессия.

Программа "Simplex" легко решает задачи минимизации суммы квадратов отклонений для линейных и нелинейных регрессий. В отличие от методов с производными, она устойчива и относительно нетребовательна к выбору начального приближения даже в случае сложных нелинейных моделей. Важно лишь не попасть в ложный (локальный) минимум, если таковой существует. Кроме того, чтобы не происходило арифметических авостов при наличии таких функций как показательная,

квадратный корень, логарифм и т.п., надо позаботиться, чтобы переменные не выходили из области допустимых значений. И даже в этом случае можно применить программу в варианте минимизации с ограничениями (см. пример ниже). Недостатком методов без производных является то, что без специальных приемов с их помощью возможно лишь точечное оценивание параметров, т.е. без оценки их достоверности.

Рассмотрим линейную регрессию вида $y(x) = ax + b$, где x – независимая переменная, $y(x)$ – наблюдения, a и b – оцениваемые параметры. Исходные данные представлены в табл. 1.

Табл. 1. Линейная регрессия, исходные данные.

u	x(u)	y(u)
1	10	12.05
2	10	11.48
3	20	12.59
4	20	15.82
5	30	17.35
6	30	16.42
7	40	16.24
8	40	17.04
9	50	20.56
10	50	19.28

Заносим на вкладке «Константы» m→2; n→10; mnk→True. На вкладке "МНК" заполняем второй и третий столбцы регрессионной таблицы. На вкладке "Simplex" начальное приближение – любое. Записываем процедуры, приняв, что a→P(1); b→P(2); x(u) →X(u); y(u) →Y(u); $y(x)$ →F1:

```
Sub OF()
F2 = 0
   For u = 1 To n:  Call MF(u): F2 = F2 + (Y(u) - F1) ^ 2:  Next u
   i1 = i1 + 1
End Sub
Sub MF(u)
   F1 = P(1) * X(u) + P(2)
End Sub
```

Запускаем макрос и получаем из точки $(0,\ 0)^T$ оценки: $\vec{p}^{\,*} = \left(0.18744,\ 10.25985\right)^T$; $F\left(\vec{p}^{\,*}\right) = 11.9716$; i1 = 89. Открываем вкладку "МНК" и просматриваем результат подгонки регрессии.

5^0. Использование программы в качестве МНК. Нелинейная регрессия.

В случае нелинейных регрессий порядок действий остаётся прежним, изменяются лишь: запись функции-модели в Sub MF(u), константы m и n, числа в регрессионной таблице.

Пусть требуется найти МНК-оценки параметров двухстадийной химической реакции типа A → B → C. Функция-модель для описания зависимости концентрации промежуточного продукта от времени x имеет вид[4]:

$$y = f(\vec{p}, x) \equiv \frac{p_1}{p_1 - p_2}\left(e^{-p_2 x} - e^{-p_1 x}\right).$$

Исходные данные для расчёта приведены в табл.2.

[4] Этот пример заимствован из [2], задача 3.44.

Табл. 2. Нелинейная регрессия, исходные данные.

u	x_u	y_u
1	0,5	0,263
2	1,0	0,455
3	1,5	0,548

Записываем процедуру для функции-модели, приняв, что p_1→P(1); p_2→P(2); x_u →X(u); y_u →Y(u); y →F1:

```
Sub MF(u)
F1 = P(1) / (P(1) - P(2)) * (Exp(-P(2) * X(u)) - Exp(-P(1) * X(u)))
End Sub
```

Для Sub OF() запись остаётся прежней, как в предыдущем примере. В качестве констант вводим: m → 2; n → 3; mnk → True. Начальные оценки можно взять $(2, 1)^T$ или иные, но нельзя в данном примере задать одинаковые. Запускаем макрос и при eps = 10^{-8} получаем: $\vec{p}^* = (0.66273, 0.15415)^T$; $F(\vec{p}^*) = 1.718_{10} - 4$ (i1 = 65). Это с большой точностью совпадает с результатом вычислений нелинейным МНК по Гауссу-Зайделю (линеаризацией модели).

6^0. Минимизация функции с ограничениями.

Этот класс задач в прикладном нелинейном программировании считается сложным [2]. Однако программа "Simplex", как показал опыт, в большинстве случаев неплохо с ним справляется. Рассмотрим одну такую задачу, при этом воспользуемся приёмом, известным как метод штрафных функций (Penalty Functions).

Пусть требуется найти минимум функции $F = F(\vec{p})$ при наличии следующих условий (ограничений):

$$h_l(\vec{p}) = 0, \quad l = 1,...,r$$
$$g_l(\vec{p}) \geq 0, \quad l = r+1,...,q.$$

Определяется "штраф" к $F = F(\vec{p})$ как мера невыполнения поставленных условий:

$$T(\vec{p}) = \left[\sum_{l=1}^{r} h_l^2(\vec{p}) + \sum_{l=r+1}^{q} U_l \cdot g_l^2(\vec{p}) \right]^{1/2},$$

где U_l – т.н. оператор Хэвисайда, такой, что он равен 1, если условие $g_l(\vec{p}) \geq 0$ нарушено, и равен 0 в противном случае. Решение задачи сводится к минимизации штрафной функции (ШФ) вида

$$\Phi(\vec{p}) = F(\vec{p}) + w \cdot T(\vec{p}),$$

где w – *вес* штрафа. Чем больше величина w, тем больше требования к выполнению ограничений, но одновременно поиск минимума всё более осложняется из-за роста "овражности" функции. Поэтому процедуру минимизации лучше осуществлять поэтапно: сначала найти минимум ШФ при w = 1, потом вес увеличить, например, в 10 раз и найти уточнённый минимум, и т.д. до получения сходящихся результатов.

Для примера рассмотрим следующую задачу. Найти минимум функции $F(\vec{x}) = -x_1^2$ при следующих ограничениях:

$$h(\vec{x}) = x_1^2 + x_2^2 - 25 = 0; \quad g_1(\vec{x}) = x_1 \geq 0; \; g_2(\vec{x}) = x_2 \geq 0$$

(правильный ответ: $x_1^* = 5; \; x_2^* = 0; \; f(\vec{x}^*) = -25$).

Задаём ЦФ в виде штрафной функции:

$$\Phi(\vec{x}) = -x_1^2 + w \cdot \left[(x_1^2 + x_2^2 - 25)^2 + U_1 x_1^2 + U_2 x_2^2 \right]^{1/2}$$

Принимаем: x_1→P(1); x_2→P(2); w→W; $h(\vec{p})$→H; $g_1(\vec{p})$→ G1; $g_2(\vec{p})$→ G2; $T(\vec{p})$→ T; $\Phi(\vec{p})$→ F2; U_1 →U1; U_2→ U2 и записываем процедуру в следующем виде:

```
Sub OF() 'задача минимизации с ограничениями
'объявляем дополнительные переменные:
Dim H, G1, G2, T As Double, W As Single, U1, U2 As Integer
```

```
H = P(1) ^ 2 + P(2) ^ 2 – 25: G1 = P(1): G2 = P(2)
If G1 < 0 Then U1 = 1 Else U1 = 0
If G2 < 0 Then U2 = 1 Else U2 = 0
W = 1: T = Sqr(H ^ 2 + U1 * G1 ^ 2 + U2 * G2 ^ 2)
F2 = -P(1) ^ 2 + W * T: i1 = i1 + 1
End Sub
```

Результаты вычислений при разных весах штрафа и старте всякий раз из точки $(0, 0)^T$ приведены в табл. 3 (eps = 10^{-8}).

Табл. 3. Результаты минимизации функции с ограничениями.

Вес штрафа, W	Достигнутые значения в точке минимума			i1
	P(1)	P(2)	F2	
1	5.6233	9.620_{10}-5	-25.0000	61
10	5,0000	5.942_{10}-4	-25.0000	625
100	5,0000	4.453_{10}-3	-25.0000	5101

Видно, что объём вычислений резко увеличивается по мере роста W, но уже при W=10 получается вполне точный результат.

7^0. Задача большой размерности.

Известно, что по мере увеличения размерности ЦФ (роста числа её параметров) поиск экстремумов всё более и более усложняется, и это происходит не только потому, что естественно увеличивается объём вычислений, но и начинают проявляться различные недостатки алгоритмов (т.н. "проклятие размерности").

В этом отношении алгоритм Нелдера-Мида также имеет свои ограничения. Считается, например, что он эффективен при *m* не более 6-7. Рассмотрим, как он проявит себя в задаче минимизации ЦФ такого вида:

$$F(\vec{p}) = \sum_{l=1}^{m} l \cdot \exp(p_l - l)^2, \quad (\#)$$

Для нее $\vec{p}^* = (1, 2, ..., m)^T$; $F^* = (m+1)m/2$. Процедура для ЦФ выглядит так:

```
Sub OF() 'задача большой размерности
Dim l as integer 'вспомогательный индекс
F2 = 0
For l = 1 to m:  F2 = F2 + l * Exp ( ( P (l) – l ) ^ 2): Next l
i1 = i1 + 1
End Sub
```

Сразу же отметим, что результат сильно зависит от задания минимального объема ДМ: чем меньше этот параметр (eps), тем успешнее алгоритм справляется с задачей, но объем вычислений возрастает. Выясняется, что при *m* вплоть до 9 (старт из начала координат) поиск минимума функции (#) не вызывает затруднений: ответ верен с точностью до 4-х знаков после запятой (i1 = 5857), однако уже при *m* = 10 процедура "выклинивается", не достигнув минимума. Подобное непредвиденное окончание процедуры происходит, как принято считать в подобных случаях, из-за *коллапса* деформируемого многогранника. Вероятно, когда центроид ДМ попадает в самое русло оврага или очень близко к нему, а прочие точки расположились "неудачно" то операции сжатия или редукции не приводят к успеху, т.к. лучшей точки всё равно не находится. Однако при замене начальной точки на достигнутую и запуска программы "с нуля", ответ в рассматриваемом случае получается правильным. При *m* = 12 ситуация ещё больше осложняется, и начальное приближение приходится заменять дважды. При m = 15 задача, как ни странно, все еще успешно решается, но уже за 3 прогона (при бóльших m программа не испытывалась, пользователь может поэкспериментировать самостоятельно). Именно для случая "выклинивания" процесса поиска предусмотрена процедура Sub RS(), упомянутая выше. При её запуске

(<Ctrl+r>) координаты достигнутого (предположительно ложного) минимума переписываются в ячейки начального приближения, и автоматически запускается Sub Siplex. Как это выглядит на практике решения рассматриваемой задачи, показывают снимки экрана (рис. 2 – 4).

Таким образом, в случае ЦФ большой размерности к результату, полученному за одну попытку (прогон), следует отнестись с осторожностью и убедиться, путём варьирования начального приближения, не найден ли локальный экстремум. Полезно также запуском процедуры Sub RS() выяснить, не "выклинилась" ли программа из-за коллапса многогранника.

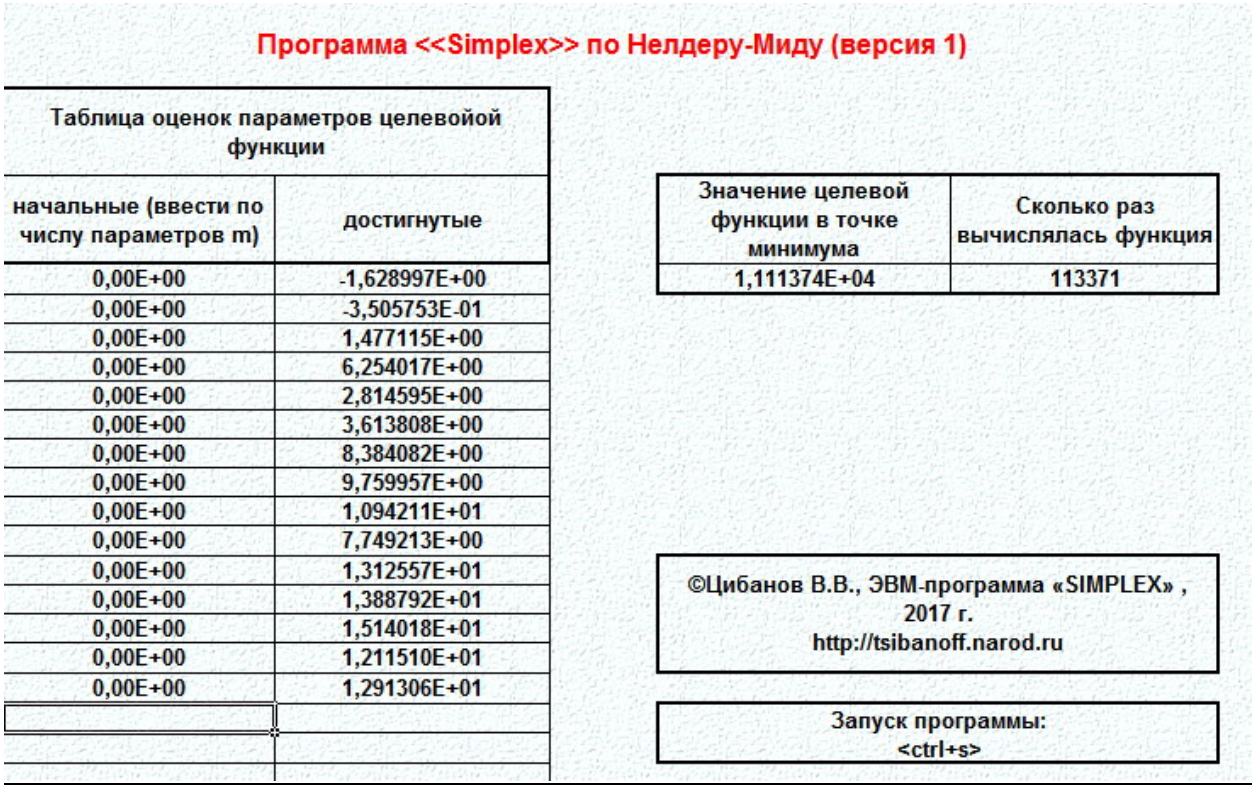

Рис. 2. Задача большой размерности. Функция (#), m = 15. Первый прогон.

Программа <<Simplex>> по Нелдеру-Миду (версия 1)

Таблица оценок параметров целевойой функции	
начальные (ввести по числу параметров m)	достигнутые
-1,63E+00	1,026861E+00
-3,51E-01	1,825290E+00
1,48E+00	3,045232E+00
6,25E+00	3,974821E+00
2,81E+00	5,052269E+00
3,61E+00	5,959315E+00
8,38E+00	7,010222E+00
9,76E+00	7,999205E+00
1,09E+01	9,006381E+00
7,75E+00	9,999039E+00
1,31E+01	1,101381E+01
1,39E+01	1,200332E+01
1,51E+01	1,300256E+01
1,21E+01	1,400586E+01
1,29E+01	1,501401E+01

Значение целевой функции в точке минимума	Сколько раз вычислялась функция
1,201019E+02	6283

©Цибанов В.В., ЭВМ-программа «SIMPLEX» , 2017 г.
http://tsibanoff.narod.ru

Запуск программы:
<ctrl+s>

Рис. 3. Задача большой размерности. Функция (#), m = 15. Второй прогон.

Программа <<Simplex>> по Нелдеру-Миду (версия 1)

Таблица оценок параметров целевойой функции	
начальные (ввести по числу параметров m)	достигнутые
1,03E+00	9,997507E-01
1,83E+00	2,000019E+00
3,05E+00	3,000037E+00
3,97E+00	4,000088E+00
5,05E+00	5,000054E+00
5,96E+00	5,999984E+00
7,01E+00	7,000006E+00
8,00E+00	7,999947E+00
9,01E+00	8,999979E+00
1,00E+01	9,999958E+00
1,10E+01	1,099997E+01
1,20E+01	1,200004E+01
1,30E+01	1,299995E+01
1,40E+01	1,400004E+01
1,50E+01	1,499999E+01

Значение целевой функции в точке минимума	Сколько раз вычислялась функция
1,200000E+02	1010

©Цибанов В.В., ЭВМ-программа «SIMPLEX» , 2017 г.
http://tsibanoff.narod.ru

Запуск программы:
<ctrl+s>

Рис. 4. Задача большой размерности. Функция (#), m = 15. Третий прогон.

Литература

1. Nelder J.A., Mead R. // Computer Journal. – 1964. – V. 7. – P. 308.
2. Химмельблау Д. Прикладное нелинейное программирование. – М.: Мир, 1975. 534 с.
3. Э. Бунин. Excel Visual Basic для приложений (серия "Без проблем!"): Пер. с англ. – М.: Восточная Книжная Компания, 1966. – 352 с.
4. URL: http://tsibanoff.narod.ru/algorythms/simplex-v1.xls. Посещен 27.10.2017.