Министерство образования Российской Федерации
РОСТОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

С.С.Михалкович, А.В.Олифер, А.М.Столяр

ЧИСЛЕННЫЕ МЕТОДЫ

Выпуск III

Оптимизация. Системы нелинейных уравнений.

Методические указания к выполнению
индивидуальных заданий на ЭВМ
для студентов 2 курса физического факультета

Ростов-на-Дону
2000

Введение

Настоящие методические указания являются продолжением [1],[2] и содержат изложение теоретического материала, а также индивидуальные задания по следующим темам курса “Численные методы”: одномерная и многомерная оптимизация, решение систем нелинейных уравнений.

Современные математические пакеты типа Maple, Mathematica, Matlab предоставляют ряд готовых процедур для решения задач этих типов. Однако, для ясного понимания сути заложенных в них численных методов, их характеристик и ограничений, на наш взгляд требуется умение реализовывать простейшие из них.

Для реализации методов можно использовать как транслирующие языки программирования типа Паскаль, С, так и интерпретирующие языки, заложенные в сами математические пакеты. При выборе среды программирования необходимо принимать в расчет следующие факторы. Система Maple, в отличие от языков Паскаль и С, поддерживает символьные вычисления, вычисления со сверхвысокой точностью, возможность быстрой визуализации исходных данных и полученных результатов. Язык программирования Maple содержит набор универсальных структур данных типа множеств и списков. Maple позволяет также сравнить полученное численное решение с решением, даваемым стандартными процедурами. Наконец, лист Maple является удобным средством отчета, поскольку содержит как тексты программ, так и результаты расчетов вместе с их графической интерпретацией. Однако, Maple-программа, не вызывающая стандартные математические процедуры, а содержащая лишь управляющие конструкции языка, уступает по скорости (нередко существенно) соответствующим Паскаль- или С-программам. Поэтому при реализации емких по времени алгоритмов рекомендуется использовать языки транслирующего типа, применяя Maple лишь как средство визуализации результатов, выбора начального приближения и т.п.

Все алгоритмы, приводимые в настоящих методических указаниях, являются простыми. Их рекомендуется выполнять полностью в Maple, концентрируясь на сути применяемых численных методов.

1 Одномерная оптимизация

1.1 Задание

Найти все локальные минимумы и максимумы функции $y=f(x)$ и точки, в которых они достигаются:

a) методом Ньютона;

b) методом золотого сечения.

Определить, сколько итераций потребуется в каждом методе для достижения точности $\varepsilon = 10^{-3}, 10^{-5}, 10^{-7}$ при одних и тех же начальных приближениях.

Включить в отчет ответы на вопросы, сформулированные в пункте “Тестовый пример”.

Рекомендуемая литература: [4], с.407–421; [6].

1.2 Тестовый пример

Для функции $F(x) = x^3 - 8x^2 + 2x - 5 + \sin x$

$$x_{\min} = 5.17574239, \quad F(x_{\min}) = -71.20016030691437 ;$$

$$x_{\max} = 0.19334459, \quad F(x_{\max}) = -4.712997997082332 .$$

В методе Ньютона определите, при каких начальных приближениях итерационный процесс сходится к минимуму, а при каких – к максимуму. Проследите, как ведет себя итерационный процесс при задании начального приближения в окрестности точки перегиба, вычисляемой в системе Maple с помощью команды **`evalf(solve(DDF(x)=0))`**.

В методе золотого сечения определите, к чему сходится итерационный процесс, если вызвать процедуру **`gold`** (см. пункт 1.5.2) с интервалом [*a*,*b*], содержащим как точку минимума, так и точку максимума.

1.3 Вводные замечания

Под *задачей оптимизации* будем понимать нахождение экстремума (минимума или максимума) функции одной или нескольких вещественных переменных. К решению оптимизационных задач сводятся задачи поиска корней нелинейных уравнений и систем, аппроксимации функций и др.

Пусть дана вещественная функция n вещественных переменных $F(\boldsymbol{x})$, $\boldsymbol{x} = (x_1, ..., x_n)$, определенная в замкнутой области $S \in R^n$. Требуется найти

$$\min_{\boldsymbol{x} \in S} F(\boldsymbol{x}) \quad \left(\max_{\boldsymbol{x} \in S} F(\boldsymbol{x}) \right) \tag{1}$$

и точку $\boldsymbol{x}^* \in S$, в которой он достигается.

Заметим, что поиск максимума функции $F(\boldsymbol{x})$ эквивалентен поиску минимума функции $-F(\boldsymbol{x})$. Поэтому далее сконцентрируем внимание именно на задаче минимизации.

Множество S выражает ограничения задачи. Например, во многих физических задачах требуется, чтобы переменные были неотрицательны: $S = \{\boldsymbol{x}: x_i \geq 0\}$. Если S совпадает с R^n, то говорят о задаче *безусловной оптимизации*, в противном случае говорят об *условной оптимизации*.

Определим также следующие два термина. Будем говорить, что $F(\boldsymbol{x})$ имеет в точке $\boldsymbol{x}^* \in S$ *глобальный минимум*, если $F(\boldsymbol{x}) \geq F(\boldsymbol{x}^*)$ для любого $\boldsymbol{x} \in S$. Аналогично, $F(\boldsymbol{x})$ имеет в точке $\boldsymbol{x}^* \in S$ *локальный минимум*, если $F(\boldsymbol{x}) \geq F(\boldsymbol{x}^*)$ для всех $\boldsymbol{x} \in S \cap D(\boldsymbol{x}^*)$, где $D(\boldsymbol{x}^*)$ – некоторая окрестность точки $\boldsymbol{x}^*$.

Если локальный минимум достигается во внутренней точке $\boldsymbol{x}^*$ области S и функция $F(\boldsymbol{x})$ дифференцируема в этой точке, то $\boldsymbol{x}^*$ является критической точкой, т.е. выполняются условия:

$$\frac{\partial F}{\partial x_i}(\boldsymbol{x}^*) = 0, \quad i = 1,...,n. \tag{2}$$

Обратное неверно: критическая точка не обязана быть экстремальной. Например, в одномерном случае для функции $y = x^3$ точка x=0 является критической, но не является точкой экстремума. В двумерном случае для функции $F(x, y) = x^2 - y^2$ критической является точка (0,0). Она также не реализует экстремум, поскольку одномерная функция $F(x,0)$ имеет минимум при $x = 0$, а одномерная функция $F(0, y)$ – максимум при $y = 0$.

Если же локальный минимум достигается на границе области S, то точка минимума не обязана быть критической. Например, функция $f(x) = x$ достигает на отрезке [1,2] минимум в точке x*=1, но $f'(1) = 1 \neq 0$.

Мы ограничимся поиском локальных минимумов в задаче безусловной оптимизации.

Опишем два основных алгоритма одномерной оптимизации. *Метод золотого сечения* использует для своей работы только значения функции, сходится всегда, но обеспечивает лишь линейную скорость сходимости ([1], с.15). *Метод Ньютона* использует значения первой и второй производных функции $F(x)$, имеет квадратичную скорость сходимости, но может расходиться при плохом выборе начального приближения. Обычно в реальных задачах комбинируют оба метода, применяя для локализации корней метод золотого сечения, а затем на заключительном этапе – метод Ньютона.

1.4 Метод Ньютона

1.4.1 Идея метода

В основе метода Ньютона лежит приближение $F(x)$ квадратичной функцией, экстремум которой находится явно и используется в качестве следующего начального приближения.

Зададимся некоторым начальным приближением x_0 и выпишем первые три члена ряда Тейлора функции $F(x)$ в окрестности этой точки:

$$F(x_0+\Delta x)\approx F(x_0)+F'(x_0)\cdot\Delta x+\frac{F''(x_0)}{2}(\Delta x)^2. \qquad (3)$$

Экстремум квадратичной функции в правой части (3) достигается при $\Delta x=-F'(x_0)/F''(x_0)$. Поэтому в качестве следующего приближения можно выбрать точку $x_1=x_0-F'(x_0)/F''(x_0)$. Получаем итерационный процесс

$$x_{k+1}=x_k-F'(x_k)/F''(x_k),\ k=0,1,\dots\ , \qquad (4)$$

дающий алгоритм *метода Ньютона*.

Отметим, что при выборе начального приближения достаточно близко к точке экстремума метод Ньютона гарантированно сходится. Отметим также, что метод Ньютона может сходиться как к локальному минимуму, так и к локальному максимуму, поэтому в полученной точке экстремума x^* требуется дополнительно вычислить $F''(x^*)$. Если $F''(x^*)>0$, то мы имеет дело с локальным минимумом, если же $F''(x^*)<0$ – с локальным максимумом. Наконец, заметим, что если для некоторого x_k окажется $F''(x_k)=0$, то метод Ньютона завершится делением на 0.

1.4.2 Замечания по численной реализации

Так как в точке экстремума x^* выполняется условие $F'(x^*)=0$, то в ее окрестности

$$\Delta F=F(x^*+\Delta x)-F(x^*)\approx\frac{F''(x^*)}{2}(\Delta x)^2. \qquad (5)$$

Поэтому, если величина $F''(x^*)$ порядка единицы, то, например, при погрешности вычисления функции $\Delta F\cong 10^{-12}$ погрешность решения Δx составит $\cong 10^{-6}$. Таким образом, при использовании типа **`real`** в языке Turbo Pascal не рекомендуется задавать погрешность вычислений $\varepsilon<10^{-6}$. В системе Maple по умолчанию количество значащих цифр при работе с вещественными числами равно 10, поэтому для вычисления решения с погрешностью $\cong 10^{-6}$ необходимо выполнить присваивание

```
Digits:=12;
```

где **Digits** – переменная среды Maple.

Процедура метода Ньютона должна иметь следующие параметры:

x0 – начальное приближение;

eps – точность;

niter – количество итераций (выходной параметр).

Найденное значение корня должно быть возвращаемым значением процедуры (в Maple процедуры могут возвращать значения, в Паскале для этого используются функции). Желательно передавать также функцию $F(x)$ в качестве функционального параметра. Наконец, если метод Ньютона расходится, то для распознавания такой ситуации можно возвращать **niter=0**.

Дадим несколько рекомендаций по работе в системе Maple.

1) Несколько связанных команд лучше набирать в одной *командной строке* Maple (она начинается с символа ">"). Если команды разделяются символом ":", то промежуточные результаты не отображаются. При нажатии в конце строки комбинации клавиш Shift-Enter создается новая строка текста без перехода к следующей командной строке.

2) Функции, задаваемые выражением, описываются в Maple либо с помощью *функционального оператора*:

```
F:=x->x^3-x^2;
```

либо с помощью команды **unapply**:

```
F:=unapply(x^3-x^2,x);
```

3) Производные вычисляются с помощью оператора **D**:

```
DF:=D(F): DDF:=D(DF);
```

4) Описание процедуры в системе Maple имеет вид:

```
newton:=proc(x0,eps::numeric,F::procedure,
             niter::evaln)
  local x,n,...
  global ...
  x:=x0;
  ...
  niter:=n;
  x;
end;
```

Описание используемых локальных и глобальных переменных в процедурах Maple необязательно, но весьма желательно. Параметры, передаваемые по значению (например, **x0**), нельзя менять в теле процедуры; для этой цели следует использовать локальные переменные, которым присваиваются значения формальных параметров (**x:=x0**). Параметр, описанный как **::evaln**, вычисляется внутри процедуры, и результат

вычислений присваивается соответствующему фактическому параметру при вызове процедуры (аналог параметра-переменной в Паскале). Возвращаемым значением процедуры является значение последнего оператора в теле процедуры.

5) При необходимости выводить на экран промежуточные результаты внутри процедуры используйте функцию **`print`**:

```
print(n,x,F(x));
```

6) Во избежание зацикливания в методе Ньютона используйте смешанный оператор цикла:

```
for n from 1 to 50 while abs(x-xold)>eps do
```

1.5 Метод золотого сечения

1.5.1 Идея метода

Пусть функция $F(x)$ определена на отрезке $[a,b]$, строго убывает при $x < x*$ ($x* \in [a,b]$) и строго возрастает при $x > x*$. Такая функция называется *унимодальной* на отрезке $[a,b]$ и имеет на $[a,b]$ единственный минимум. Заметим, что унимодальная функция не обязана быть непрерывной.

Назовем интервал, на котором функция заведомо имеет минимум, *интервалом неопределенности* (ИН). Вначале ИН совпадает с отрезком $[a,b]$. Наша цель – уменьшить длину ИН до достижения заданной точности. Для этого вычислим функцию $F(x)$ в точках $x_1, x_2 \in [a,b]$, $x_1 < x_2$.

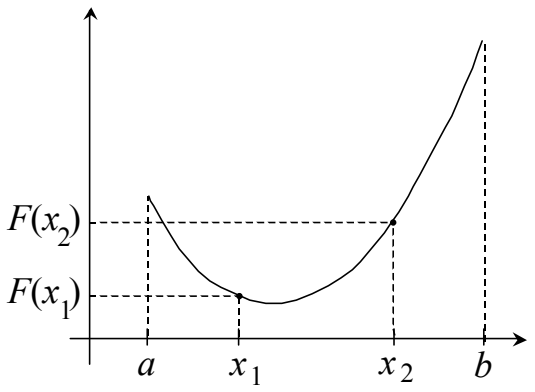

Рис. 1

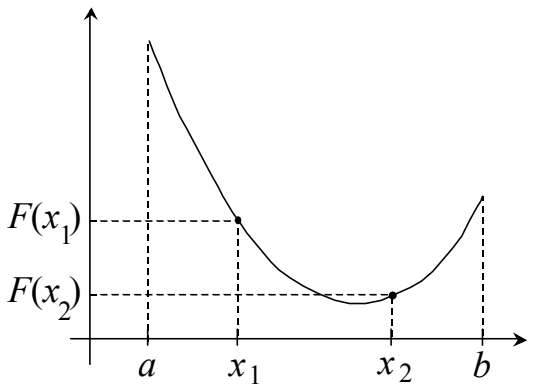

Рис. 2

Если $F(x_1) < F(x_2)$ (рис.1), то отрезок $[a, x_2]$ заведомо содержит минимум, поэтому его можно выбрать в качестве нового ИН. В противном случае (рис.2) – в качестве нового ИН выбирается отрезок $[x_1, b]$. Для каждого последующего ИН функцию достаточно вычислять в одной точке, поскольку ее значение в другой точке внутри интервала известно с предыдущего шага. Например, в случае, изображенном на рис.1, сле-

дует выбрать точку $x_3 \in [a, x_2]$ и сравнить $F(x_1)$ и $F(x_3)$. Этот процесс следует продолжать, пока длина интервала не окажется меньше наперед заданной точности ε.

Как выбирать точки x_i так, чтобы за минимальное количество итераций максимально уменьшить ИН? Метод золотого сечения предлагает следующее решение этого вопроса.

Золотое сечение отрезка

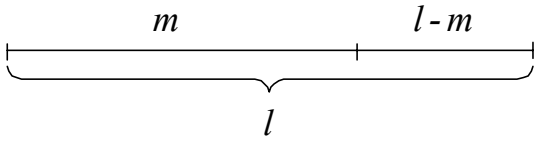

Рис. 3

Пусть у нас имеется отрезок длины *l* (рис.3). Разобьем его на части длины *m* и *l–m* (*m* > *l–m*) таким образом, что *отношение длины большей части к длине всего отрезка равно отношению длины меньшей части к длине большей*, т.е. выполняется пропорция золотого сечения:

$$\frac{m}{l} = \frac{l-m}{m} \ . \tag{6}$$

Учитывая, что $m > 0$ и $l > 0$, из (6) нетрудно получить:

$$r = \frac{m}{l} = \frac{\sqrt{5}-1}{2} \approx 0.6180 \ .$$

Число $1/r \approx 1.6180$ называется *золотым сечением* и является фундаментальным числом, возникающим во многих областях науки и техники.

В методе золотого сечения точки x_1 и x_2 выбираются так, что

$$\frac{b-x_1}{b-a} = \frac{x_2-a}{b-a} = r, \tag{7}$$

откуда $x_1 = b - r(b-a),\ x_2 = a + r(b-a)$ или $x_2 = b + a - x_1$. Нетрудно видеть, что при этом выполняются соотношения $\frac{x_1-a}{x_2-a} = \frac{b-x_1}{b-x_2} = r$, т.е. точка x_2 для отрезка $[x_1, b]$ выполняет ту же роль, что и точка x_1 для отрезка [*a*,*b*], а точка x_1 для отрезка $[a, x_2]$ – ту же роль, что и точка x_2 для [*a*,*b*].

Из (7) также следует, что длина ИН уменьшается на каждом шаге в $1/r$ раз. Таким образом, ***за пять итераций метода золотого сечения длина интервала неопределенности уменьшается в*** $(1/r)^5 \approx 11.1$ раз, т.е. ***примерно на порядок***.

1.5.2 Замечания по численной реализации

Приведем полную реализацию процедуры метода золотого сечения в системе Maple, учитывая все сделанные выше замечания.

```
> gold:=proc(a,b,eps::numeric,
             f::procedure,niter::evaln)
    local n,aa,bb,ab,xa,xb,fa,fb,fxa,fxb,r;
    global DEB;
    r:=evalf(sqrt(5)-1)/2;
    fa:=f(a); fb:=f(b);
    aa:=a; bb:=b;
    xa:=bb-r*(bb-aa); xb:=r*(bb-aa)+aa;
    fxa:=f(xa); fxb:=f(xb);
    for n from 1 to 100
      while (abs(bb-aa)>eps) do
      if (fxb<fxa) then
        aa:=xa; fa:=fxa;
        xa:=xb; fxa:=fxb;
        xb:=bb+aa-xa;
        fxb:=f(xb);
      else
        bb:=xb; fb:=fxb;
        xb:=xa; fxb:=fxa;
        xa:=bb+aa-xb;
        fxa:=f(xa);
      fi;
      ab:=(aa+bb)/2;
      if DEB=1 then print(n,ab,f(ab)); fi;
    od;
    niter:=n-1;
    ab;
  end:
```

В зависимости от значения глобальной переменной **DEB** выводятся или нет результаты промежуточных вычислений. Передача функции в качестве параметра понадобится нам при реализации многомерной оптимизации.

Заметим, что при поиске максимального значения условие **fxb<fxa** заменяется условием **fxb>fxa**. Однако, условие “ищется минимум или максимум” можно передавать в качестве параметра.

Для использования процедуры **gold** требуется, чтобы на отрезке $[a,b]$ находилась ровно одна экстремальная точка функции $F(x)$. Выбор такого отрезка называется *локализацией экстремума*. В простейших ситуациях его можно производить непосредственно по виду графика

функции $F(x)$. Однако, мы рекомендуем автоматизировать процесс локализации.

Пусть на некотором отрезке [*A*,*B*] требуется найти все экстремумы. Разобьем его точками x_i на *N* равных частей. При выполнении условий $F(x_i) > F(x_{i-1})$, $F(x_i) > F(x_{i+1})$ на отрезке $[x_{i-1}, x_{i+1}]$ находится локальный максимум, при выполнении же условий $F(x_i) < F(x_{i-1})$, $F(x_i) < F(x_{i+1})$ – локальный минимум. Число *N* следует выбирать так, чтобы длина каждой части была заведомо меньше половины расстояния между соседними локальными экстремумами (в этом случае на каждом участке монотонности функции будет как минимум две точки x_i и мы не "проскочим" минимум) и в то же время не слишком большим (для сокращения объема вычислений).

2 Многомерная оптимизация

2.1 Задание

Найти локальный минимум (максимум) функции двух переменных *F*(*x*,*y*) и точку, в которой он достигается:

а) методом покоординатного спуска;

б) методом наискорейшего градиентного спуска.

Определить, сколько итераций потребуется в каждом методе для достижения точности $\varepsilon = 10^{-3}, 10^{-5}, 10^{-7}$ при одних и тех же начальных приближениях.

В Maple построить график функции *z*=*F*(*x*,*y*) и график ее линий уровня. При реализации в системе Maple на последнем графике построить также ломаную, соединяющую все последовательные приближения к решению.

Рекомендуемая литература: [3], с.290, 292–293, 329–337;
[4], с. 424–427; [6].

2.2 Тестовый пример

$f(x, y) = x^2 + 2x + y^2 - \sin(xy)$;
$\min f(x, y) = -1.274688296$;
$(x, y)_{\min} = (-1.2053533, -0.4975176)$.

2.3 Методы покоординатного и наискорейшего спуска

2.3.1 Идея методов

Основная идея методов спуска состоит в том, что при заданном начальном приближении определяется направление, в котором функция убывает, и производится перемещение в этом направлении. Если величина шага перемещения выбрана не очень большой, то значение функции уменьшится.

Алгоритм любого метода спуска дается формулой:

$$\boldsymbol{x}^{k+1} = \boldsymbol{x}^k + h_k \boldsymbol{v}^k,$$

где $\boldsymbol{x}^k$ – k-тое приближение к решению, $\boldsymbol{v}^k$ – некоторый вектор, h_k – длина шага в направлении $\boldsymbol{v}^k$.

Как правило, в качестве h_k выбирается значение h, реализующее

$$\min_h F(\boldsymbol{x}^k + h\boldsymbol{v}^k). \qquad (8)$$

В методе *наискорейшего градиентного спуска* (или просто *наискорейшего спуска*) вектор $\boldsymbol{v}^k$ полагается равным вектору, противоположному градиенту функции F в точке $\boldsymbol{x}^k$ и задающему направление наискорейшего убывания функции F в точке $\boldsymbol{x}^k$:

$$\boldsymbol{v}^k = -grad\, F(\boldsymbol{x}^k) = -\left(\frac{\partial F}{\partial x_1}(\boldsymbol{x}^k),\ldots,\frac{\partial F}{\partial x_n}(\boldsymbol{x}^k)\right).$$

В методе *покоординатного спуска* в качестве $\boldsymbol{v}^k$ выбираются векторы, параллельные поочередно каждой из осей координат (в двумерном случае $\boldsymbol{v}^{2k} = (1,0)$, $\boldsymbol{v}^{2k+1} = (0,1)$, $k = 0,1,2,\ldots$).

Таким образом, задача многомерной минимизации сводится на каждом шаге к решению задачи (8). Одномерная минимизация по формуле (8) проводится, как правило, методом золотого сечения с учетом следующего замечания. На прямой $\boldsymbol{x}^k + h\boldsymbol{v}^k$ может существовать несколько локальных минимумов. Поскольку нас интересует ближайший, то до применения метода золотого сечения его вначале локализуют способом, аналогичным описанному в методе золотого сечения. Отличие состоит в том, что, задавшись некоторым шагом h_0, заведомо меньшим половины расстояния между двумя локальными экстремумами исследуемой функции, мы вычисляем значения $F_j^k = F(\boldsymbol{x}^k + h_0 j\, \boldsymbol{v}^k)$, $j = 0,1,2,\ldots$, до тех пор, пока не выполнится условие $F_{j+1}^k > F_j^k$. После этого вызываем процедуру метода золотого сечения, в которой миними-

зируется функция $F(\boldsymbol{x}^k + h\boldsymbol{v}^k)$ по переменной h на сегменте $[h_0(j-1), h_0(j+1)]$.

В методе покоординатного спуска неизвестно, убывает или возрастает функция в направлении $\boldsymbol{v}^k$. Поэтому здесь при выполнении на первом шаге условия $F_1^k > F_0^k$ следует изменить направление на противоположное или, что эквивалентно, изменить знак h_0 и повторить алгоритм локализации.

Метод наискорейшего спуска работает, как правило, быстрее метода покоординатного спуска, но и требует вычисления частных производных функции. В некоторых же ситуациях, когда линии уровня сильно вытянуты вдоль некоторого направления, метод градиентного спуска

Метод покоординатного спуска

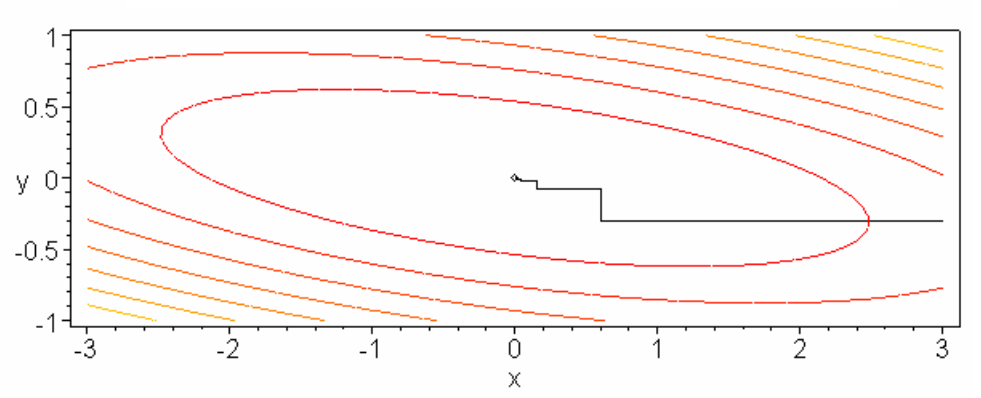

Рис. 4

Метод наискорейшего спуска

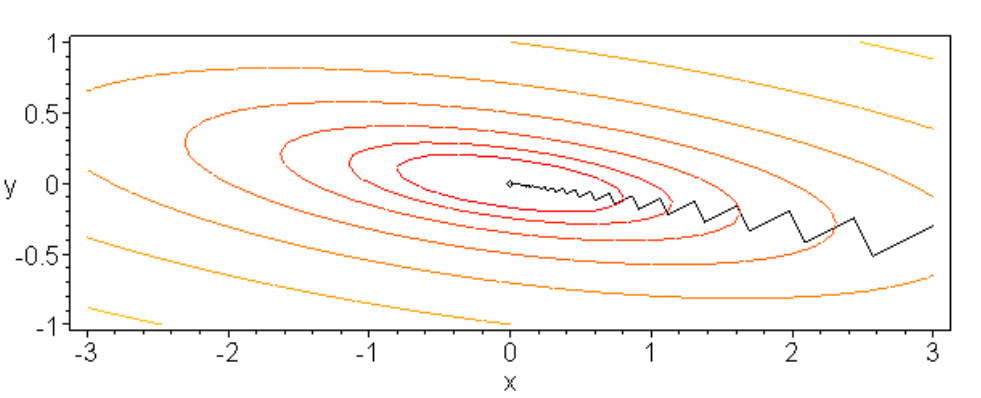

Рис. 5

неэффективен. Например, на рисунках 4 и 5 приведены результаты применения методов покоординатного и наискорейшего спусков для функции $z(x, y) = x^2 + 4xy + 16y^2$ при начальном приближении $\boldsymbol{x}^0 = (3, -0.3)$.

2.3.2 Замечания по численной реализации

Как должны соотноситься точность ε решения задачи минимизации и точность ε_1 поиска минимума функции одной переменной по формуле (8)? Мы рекомендуем самый простой путь: положить $\varepsilon_1 = \varepsilon$. Однако, следует отдавать себе отчет в том, что на первых итерациях (когда h_k – большие) искать одномерный минимум с высокой точностью – расточительство.

Процедуры наискорейшего градиентного и покоординатного спуска должны иметь следующие параметры: (x_0, y_0) – начальное приближение, F – минимизируемая функция, ε – точность решения и h – шаг локализации. При их вызове шаг локализации следует задавать не слишком маленьким (для ускорения вычислений) и не слишком большим (чтобы не “проскочить” минимум).

Приведем несколько рекомендаций по реализации в системе Maple.

1) Частные производные в системе Maple вычисляются следующим образом:

```
f:=(x,y)->...;
dfx:=D[1](f); dfy:=D[2](f);
```

2) Двумерные точки и векторы в системе Maple следует представлять двухкомпонентным списком:

```
v0:=[x0,y0];
...
GRAD:=[-dfx(v0[1],v0[2]),-dfy(v0[1],v0[2])];
```

Далее с такими величинами можно работать как с векторами:

```
v0:=v0+t0*GRAD;
```

3) Двумерные точки могут выступать также в качестве возвращаемого значения функции. Таким образом, процедуры спуска могут возвращать точку локального экстремума.

4) Расстояние между двумя точками вычисляется с помощью функции distance и требует подключение пакета student:

```
with(student):
...
v1:=[1e10,1e10]: v0:=[x0,y0]:
while distance(v0,v1)>eps do
...
```

Функцию **`distance`** естественно использовать и для нормировки вектора градиента:

```
GRAD:=GRAD/distance([0,0],GRAD);
```

5) Как использовать в процедурах спуска процедуру gold без модификации? В Maple для реализации данной идеи потребуется определить локальную функцию одной переменной, используя функцию unapply:

```
FF:=unapply(F(v0[1]+t*GRAD[1],
              v0[2]+t*GRAD[2]),t);
t0:=gold(-h,h,FF);
```

После этого новое начальное приближение находится с помощью оператора:

```
v0:=v0+t0*GRAD;
```

6) Для запоминания всех итераций метода спуска рекомендуется ввести глобальный список точек LST. Его инициализация и добавление новой точки осуществляется следующим образом:

```
LST:=[];
...
LST:=[op(LST),v0];
```

7) Для вывода на одном графике линий уровня и ломаной, соединяющей последовательные итерации метода спуска (см. рис. 4,5) использовались следующие строки:

```
with(plots):
...
R1:=3: R2:=1:
A:=contourplot(F(x,y), x=-R1..R1, y=-R2..R2,
               axes=BOXED);
B:=PLOT(CURVES(LST));
display(A,B);
```

3 Системы нелинейных уравнений

3.1 Задание

Найти все решения системы нелинейных уравнений

а) методом простой итерации;

б) методом Ньютона.

Определить, сколько итераций N потребуется в каждом методе для достижения точности $\varepsilon = 10^{-p}$ при одних и тех же начальных приближениях (p = 2,...,8 при реализации на Паскале, p = 2,...,15 при реализации в Maple). При выборе функции $\boldsymbol{g}(\boldsymbol{x})$ в методе простой итерации проверить достаточное условие сходимости.

Рекомендуемая литература: [3], с.255–257, 317–329; [4], с.298–303; [5], с.75–79; [6]

3.2 Тестовый пример

$$\begin{cases} F_1(x,y) = \dfrac{\cos y}{3} + 0.3; \\ F_2(x,y) = \sin(x - 0.6) - 1.6 \end{cases}$$

Решение: $(x,y)^* = (0.1511190, -2.0340257)$.

3.3 Метод простой итерации

3.3.1 Идея метода

Рассмотрим систему нелинейных уравнений

$$\begin{cases} F_1(x_1,...,x_n) = 0, \\ F_2(x_1,...,x_n) = 0, \\ ... \\ F_n(x_1,...,x_n) = 0, \end{cases}$$

или в векторной форме

$$\boldsymbol{F}(\boldsymbol{x}) = 0, \tag{9}$$

где $\boldsymbol{F}$ – n-мерная вектор-функция, действующая в пространстве R^n. В методе простой итерации векторное уравнение (9) преобразуется к виду

$$\boldsymbol{x} = \boldsymbol{g}(\boldsymbol{x}), \tag{10}$$

где $\boldsymbol{g}(\boldsymbol{x})$ – функция, действующая из R^n в R^n. Далее выбирается некоторое начальное приближение $\boldsymbol{x}^0$ и рассматривается итерационный процесс

$$\boldsymbol{x}^{m+1} = \boldsymbol{g}(\boldsymbol{x}^m), \tag{11}$$

Сформулируем достаточное условие его сходимости.

Пусть существует замкнутая область $D \subset R^n$ такая, что для любого $\boldsymbol{x} \in D$ $\boldsymbol{g}(\boldsymbol{x})$ также принадлежит D. Пусть также для любых $\boldsymbol{x}_1$ и $\boldsymbol{x}_2 \in D$ выполняется условие

$$\|\boldsymbol{g}(\boldsymbol{x}_1) - \boldsymbol{g}(\boldsymbol{x}_2)\| \le q\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|, \; q < 1, \tag{12}$$

где $\|\cdot\|$ – некоторая норма в R^n. Тогда нетрудно доказать, что в D существует единственное решение $\boldsymbol{x}^*$ уравнения (10) такое, что при выборе

любого $\boldsymbol{x}_0 \in D$ итерационный процесс (11) сходится к этому решению. Причем, имеет место следующая оценка скорости сходимости:

$$\left\| \boldsymbol{x}^m - \boldsymbol{x}^* \right\| \le \frac{cq^m}{1-q} \tag{13}$$

с некоторой константой *c*.

Функция $\boldsymbol{g}$, удовлетворяющая условию (12), называется *сжимающим отображением*, а решение уравнения (10) – *неподвижной точкой* функции $\boldsymbol{g}$.*

Заметим, что $\left\| \boldsymbol{x}^{m+1} - \boldsymbol{x}^* \right\| = \left\| \boldsymbol{g}(\boldsymbol{x}^m) - \boldsymbol{g}(\boldsymbol{x}^*) \right\| \le q \left\| \boldsymbol{x}^m - \boldsymbol{x}^* \right\|$, поэтому метод простой итерации имеет порядок сходимости 1.

Если функция $\boldsymbol{g}(\boldsymbol{x})$ является непрерывно-дифференцируемой в области *D*, то для выполнения условия (12) достаточно чтобы

$$\left\| \boldsymbol{g}'(\boldsymbol{x}) \right\| \le q < 1 \quad \text{для любого } \boldsymbol{x} \in D. \tag{14}$$

Здесь $\boldsymbol{g}'(\boldsymbol{x}) = \left\{ \dfrac{\partial g_i(\boldsymbol{x})}{\partial x_j} \right\}_{i,j=1}^{n}$ – матрица-функция, $\left\| \boldsymbol{g}'(\boldsymbol{x}) \right\|$ – ее норма, согласованная с нормой вектора в R^n. Наиболее употребительны следующие векторные и матричные нормы ([3], с.255):

$$\left\| \boldsymbol{x} \right\|_\infty = \max_{1 \le i \le n} \left| x_i \right|, \qquad \left\| \boldsymbol{A} \right\|_\infty = \max_{1 \le i \le n} \left(\sum_{j=1}^{n} \left| a_{ij} \right| \right),$$

$$\left\| \boldsymbol{x} \right\|_1 = \sum_{i=1}^{n} \left| x_i \right|, \qquad \left\| \boldsymbol{A} \right\|_1 = \max_{1 \le j \le n} \left(\sum_{i=1}^{n} \left| a_{ij} \right| \right).$$

Таким образом, если $\sum_{j=1}^{n} \left| \dfrac{\partial g_i(\boldsymbol{x})}{\partial x_j} \right| \le q < 1$ для любого $i = 1,\ldots,n$, $\boldsymbol{x} \in D$, то выполняется условие (14) с $\left\| \cdot \right\|_\infty$, а если $\sum_{i=1}^{n} \left| \dfrac{\partial g_i(\boldsymbol{x})}{\partial x_j} \right| \le q < 1$ для любого $j = 1,\ldots,n$, $\boldsymbol{x} \in D$, то выполняется условие (14) с $\left\| \cdot \right\|_1$.

Замечание. Функция $\boldsymbol{g}(\boldsymbol{x})$ неоднозначно определяется по функции $\boldsymbol{F}(\boldsymbol{x})$. Как практически выбрать $\boldsymbol{g}(\boldsymbol{x})$ так, чтобы выполнялось условие (14)? Если $\boldsymbol{F}(\boldsymbol{x})$ непрерывно дифференцируема и матрица-функция $\boldsymbol{F}'(\boldsymbol{x})$ невырождена в окрестности точки $\boldsymbol{x}_0$, то в общем случае можно положить

$$\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{x} - (\boldsymbol{F}'(\boldsymbol{x}_0))^{-1} \boldsymbol{F}(\boldsymbol{x}). \tag{15}$$

Действительно, уравнение (10) с выбранной таким образом функцией $\boldsymbol{g}(\boldsymbol{x})$ эквивалентно уравнению (9) (поскольку $\boldsymbol{F}(\boldsymbol{x}^*)=0$). Более того, в окрестности точки $\boldsymbol{x}_0$ выполняется приближенное равенство $\boldsymbol{F}'(\boldsymbol{x})\approx\boldsymbol{F}'(\boldsymbol{x}_0)$, поэтому $\|\boldsymbol{g}'(\boldsymbol{x})\|=\left\|E-(\boldsymbol{F}'(\boldsymbol{x}_0))^{-1}\boldsymbol{F}'(\boldsymbol{x})\right\|\le q<1$, где E – единичная матрица. Если при этом $\boldsymbol{x}^*$ попадает в эту окрестность, то выполняется условие (14). В частных случаях выбор функции $\boldsymbol{g}(\boldsymbol{x})$ и проверка условия (14) могут быть осуществлены значительно проще.

3.3.2 Проверка условий сходимости

Рассмотрим два примера, иллюстрирующие частный и общий случаи построения функции $\boldsymbol{g}(\boldsymbol{x})$ и проверки условий сходимости в пространстве R^2. Будем полагать $\boldsymbol{x}=(x,y)$.

Пример 1. Рассмотрим систему ([5], с.75-76):

$$\begin{cases} F_1(x,y)=x-\left(\dfrac{1}{3}\cos y+0.3\right)=0; \\ F_2(x,y)=y-\big(\sin(x-0.6)-1.6\big)=0. \end{cases}$$

Построим в Maple графики неявных функций $F_1(x,y)$, $F_2(x,y)$, используя вызов функции **`implicitplot`** (см. пункт 3.2.3):

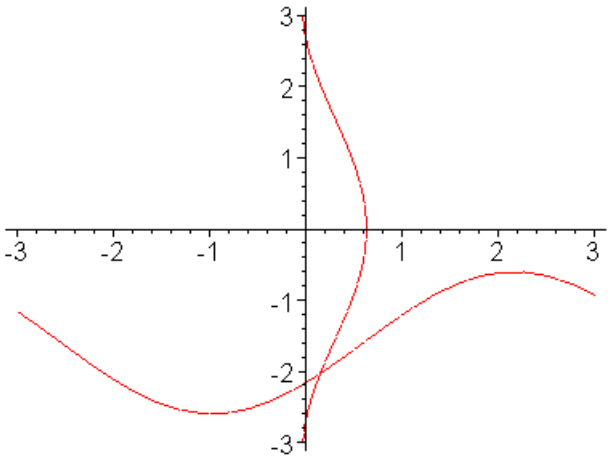

Рис. 6

Из рисунка видно, что решение локализовано в области D: $0\le x\le 0.3$; $-2.2\le y\le -1.8$.

Положим $g_1(x,y)=\dfrac{1}{3}\cos y+0.3$, $g_2(x,y)=\sin(x-0.6)-1.6$ и проверим условие (14) в области D для нормы $\|\cdot\|_\infty$:

$$\left|\frac{\partial g_1}{\partial x}\right|+\left|\frac{\partial g_2}{\partial x}\right|=\big|\cos(x-0.6)\big|\le\cos 0.3<1;$$

$$\left|\frac{\partial g_1}{\partial y}\right| + \left|\frac{\partial g_2}{\partial y}\right| = \left|-\frac{1}{3}\sin y\right| \le \frac{1}{3} < 1.$$

Таким образом, условие сходимости (14) выполняется и в качестве начального приближения $\boldsymbol{x}_0$ можно выбрать любую точку области D.

Заметим, что если второе уравнение имеет вид $F_2(x,y) = y - \frac{1}{2}(\sin(x-0.6) - 1.6) = 0$, то условие (14) выполняется для любой точки $(x,y) \in R^2$.

Пример 2. Рассмотрим систему

$$\begin{cases} F_1(x,y) = x + \cos y + y + 0.3 = 0; \\ F_2(x,y) = y - (\sin(x-0.6) + x - 1.6) = 0. \end{cases}$$

Построим в Maple графики функций $F_1(x,y)$, $F_2(x,y)$:

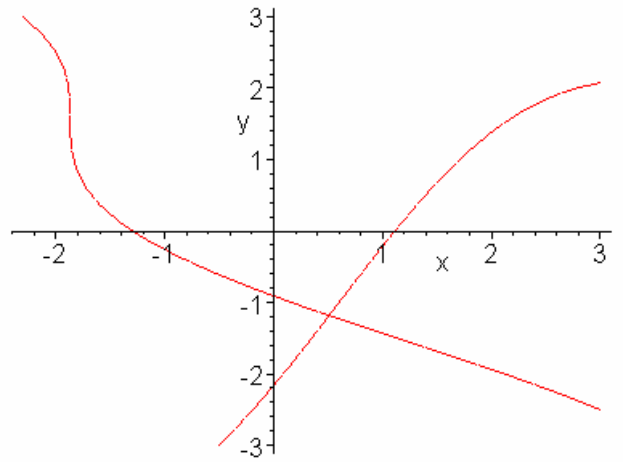

Рис. 7

Из рисунка видно, что решение локализовано в области D: $0.4 \le x \le 0.6$; $-1.1 \le y \le -1.3$.

Попытаемся выбрать $g_1(x,y) = -(\cos y + y + 0.3)$, $g_2(x,y) = \sin(x-0.6) + x - 1.6$. Нетрудно видеть, что в области D

$$\left|\frac{\partial g_2}{\partial x}\right| = \left|\cos(x-0.6) + 1\right| \ge 1 + \cos 0.2 > 1.$$

Поэтому условие (14) не выполняется ни для $\|\cdot\|_\infty$, ни для $\|\cdot\|_1$. Итерационный процесс с так выбранной функцией $\boldsymbol{g}(\boldsymbol{x})$ будет расходиться при любом выборе начального приближения (проверьте!). Чтобы получить сходящийся итерационный процесс, необходимо руководствоваться общим подходом и выбрать $\boldsymbol{g}(\boldsymbol{x})$ по формуле (15), где точка $\boldsymbol{x}_0$ достаточно близка к решению (скажем, $\boldsymbol{x}_0 = (0.5, -1.1)$). Обращение матрицы $\boldsymbol{F}'(\boldsymbol{x}_0)$ следует проводить, например, методом Крамера (см. пункт 3.4.2).

3.3.3 Замечания по численной реализации

1) При подборе начального приближения рекомендуется построить графики функций $F_1(x,y), F_2(x,y)$ на одном рисунке, используя функцию **implicitplot** из пакета **plots** в системе Maple:

```
implicitplot({x-G1(x,y),y-G2(x,y)},
  x=–3..3,y=–3..3);
```

2) Процедура метода Ньютона должна иметь следующие параметры:

x0 – начальное приближение;

eps – точность;

niter – количество итераций (выходной параметр).

Если программирование ведется в системе Maple, то вектор решения должен быть возвращаемым значением процедуры; при программировании на Паскале вектор решения необходимо возвращать как параметр-переменную процедуры.

3) Реализация метода простой итерации на языке Паскаль не должна вызвать затруднений. При программировании в системе Maple рекомендуется избрать один из следующих двух способов для работы с матрицами-функциями. Пусть имеются следующие определения функций:

```
G1:=(x,y)-> ...
G2:=(x,y)-> ...
```

В первом способе функция **G** определяется как:

```
G:=x->[G1(x[1],x[2]), G2(x[1],x[2])];
```

В этом случае формула (11) будет иметь вид:

```
xx:=evalf(G(xx));
```

Во втором способе определение **G** и вид формулы (11) следующие:

```
G:=(x,y)->[G1(x,y),G2(x,y)];
xx:=evalf(G(xx[1],xx[2]));
```

Если процедура метода простой итерации в Maple имеет следующий заголовок:

```
iter:=proc(x0::list,eps::numic,
           G::procedure,niter::evaln) ,
```

то график зависимости количества итераций от степени p погрешности $\varepsilon = 10^{-p}$ может быть построен с использованием следующего участка кода:

```
> eps:=0.1: Digits:=20:
  l:=[]:
```

```
  for m from 1 to 15 do
    x:=[0,0]:
    iter(x,eps,G,n);
    l:=[op(l),[m,n]];
    eps:=eps/10:
  od:
> plot(l, 0..15, style=point);
```

3.4 Метод Ньютона

3.4.1 Идея метода

Пусть функция $\boldsymbol{F}(\boldsymbol{x})$ в уравнении (9) непрерывно дифференцируема, $\boldsymbol{x}^*$ – решение уравнения (9), а $\boldsymbol{x}^m$ – некоторое приближение к решению. Если $\left\| \boldsymbol{x}^m - \boldsymbol{x}^* \right\|$ мала, то

$$\boldsymbol{F}(\boldsymbol{x}^*) \approx \boldsymbol{F}(\boldsymbol{x}^m) + \boldsymbol{F}'(\boldsymbol{x}^m)(\boldsymbol{x}^* - \boldsymbol{x}^m),$$

где $\boldsymbol{F}'(\boldsymbol{x}) = \left\{ \dfrac{\partial F_i(\boldsymbol{x})}{\partial x_j} \right\}_{i,j=1}^{n}$ – матрица Якоби.

Поскольку $\boldsymbol{F}(\boldsymbol{x}^*) = 0$, то приближенно

$$\boldsymbol{F}(\boldsymbol{x}^m) + \boldsymbol{F}'(\boldsymbol{x}^m)(\boldsymbol{x} - \boldsymbol{x}^m) = 0.$$

Возьмем в качестве следующего приближения $\boldsymbol{x}^{m+1}$ решение последнего уравнения. Если $\boldsymbol{F}'(\boldsymbol{x}^m)$ – обратимая матрица, то

$$\boldsymbol{x}^{m+1} = \boldsymbol{x}^m - \left(\boldsymbol{F}'(\boldsymbol{x}^m)\right)^{-1} \boldsymbol{F}(\boldsymbol{x}^m). \qquad (16)$$

Данный итерационный процесс называют *методом Ньютона решения систем нелинейных уравнений*.

Пусть в некоторой окрестности D решения $\boldsymbol{x}^*$ $\boldsymbol{F}'(\boldsymbol{x})$ существует и ограничена по норме и пусть также выполняется условие

$$\left\| \boldsymbol{F}(\boldsymbol{x}_1) - \boldsymbol{F}(\boldsymbol{x}_2) - \boldsymbol{F}'(\boldsymbol{x}_2)(\boldsymbol{x}_1 - \boldsymbol{x}_2) \right\| \le c \left\| \boldsymbol{x}_1 - \boldsymbol{x}_2 \right\|^2 \qquad (17)$$

для любых $\boldsymbol{x}_1, \boldsymbol{x}_2 \in D$. Тогда при выборе начального приближения $\boldsymbol{x}^0$ достаточно близко к $\boldsymbol{x}^*$ последовательность $\boldsymbol{x}^m$, определяемая формулой (15), сходится к решению $\boldsymbol{x}^*$ [3, с.324].

Можно показать, что $\left\| \boldsymbol{x}^{m+1} - \boldsymbol{x}^* \right\| \le c \left\| \boldsymbol{x}^m - \boldsymbol{x}^* \right\|^2$ для любого m, т.е. многомерный метод Ньютона, как и одномерный, имеет квадратичный порядок сходимости.

На практике метод Ньютона гарантированно сходится лишь в достаточно малой окрестности решения, поэтому его обычно применяют для быстрого уточнения корней.

Заметим также, что метод Ньютона является разновидностью метода простой итерации с $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{x} - \left(\boldsymbol{F}'(\boldsymbol{x})\right)^{-1}\boldsymbol{F}(\boldsymbol{x})$.

3.4.2 Замечания по численной реализации

1) Процедура метода Ньютона должна иметь те же параметры, что и процедура метода простой итерации.

2) При реализации метода Ньютона на языке Паскаль можно не вычислять всякий раз обратную матрицу $\left(\boldsymbol{F}'(\boldsymbol{x}^m)\right)^{-1}$ из (16), а вычислять вектор $h = \left(\boldsymbol{F}'(\boldsymbol{x}^m)\right)^{-1}\boldsymbol{F}(\boldsymbol{x}^m)$ как решение системы уравнений $\boldsymbol{F}'(\boldsymbol{x}^m)h = \boldsymbol{F}(\boldsymbol{x}^m)$. Поскольку в заданиях размерность системы n=2, то для решения такой системы без существенной потери эффективности можно использовать теорему Крамера:

$$h_i = \frac{\Delta_i}{\Delta},\ i=1..2,\ \Delta = \left|\boldsymbol{F}'(\boldsymbol{x}^m)\right| = \begin{vmatrix} (F_1')_{x_1}(\boldsymbol{x}^m) & (F_1')_{x_2}(\boldsymbol{x}^m) \\ (F_2')_{x_1}(\boldsymbol{x}^m) & (F_2')_{x_2}(\boldsymbol{x}^m) \end{vmatrix},$$

$$\Delta_1 = \begin{vmatrix} F_1(\boldsymbol{x}^m) & (F_1')_{x_2}(\boldsymbol{x}^m) \\ F_2(\boldsymbol{x}^m) & (F_2')_{x_2}(\boldsymbol{x}^m) \end{vmatrix},\ \Delta_2 = \begin{vmatrix} (F_1')_{x_1}(\boldsymbol{x}^m) & F_1(\boldsymbol{x}^m) \\ (F_2')_{x_1}(\boldsymbol{x}^m) & F_2(\boldsymbol{x}^m) \end{vmatrix}.$$

3) В системе Maple лучше вычислить матрицу Якоби и затем ее обратить аналитически, используя функции **jacobian** и **inverse** из пакета **linalg**. Если вектор-функция F задается командой **F:=(x,y)->[F1(x,y),F2(x,y)]**, то в процедуре метода Ньютона для этого следует выполнить следующие действия:

```
FP:=jacobian(F(x,y),[x,y]):
FPINV:=inverse(FP):
```

Приведем также вариант основного цикла метода Ньютона:

```
> v:=xx; v1:=[1e10,1e10];
  for n to 10 while distance(v,v1)>eps do
    v1:=eval(v);
    M:=eval(eval(FPINV),[x=v[1],y=v[2]]):
    v:=evalm(v-M&*F(v[1],v[2]));
  od;
```

Здесь **xx** – вектор начального приближения, функция **eval** необходима для полного вычисления символьных выражений в процедуре, функция **evalm** предназначена для проведения матричных вычислений, операция **&*** означает матрично-векторное произведение.

Варианты заданий к теме “Одномерная оптимизация”

1. $F(x) = \arctan(x-2) - \frac{x}{3}$
2. $F(x) = (x+3)\,e^{-\frac{1}{x^2-2x+2}}$
3. $F(x) = x - 5\sin\frac{1}{(x-1)^2+1}$
4. $F(x) = \ln(x^2+3x+3)\,(\ln(x^2+3x+4)-4)$
5. $F(x) = e^x + e^{-x-1} - 3x^2 + 1$
6. $F(x) = 0.01x + \cos(\arctan(x^2-2x+3) + 1.9$
7. $F(x) = \cos(\arctan(x^2-2x+3) + 1.9$
8. $F(x) = e^{0.05x-1} + \frac{x}{(x-1)^2+1}$
9. $F(x) = \frac{x^2+1}{x^2+2x+2(e^x+e^{-x})}$
10. $F(x) = \frac{x^4+1}{(x-1)^4+20e^{-x^2}}$
11. $F(x) = 2^{-x^2+3x} + 2^{-5x^2-2x+1}$
12. $F(x) = \ln(x^2+2x+1.1)\ln(x^2+3x+3)$
13. $F(x) = (x-\arctan(x+1))^3 - 4x^2 + 3x + 10$
14. $F(x) = \left(x-\arctan(x+1)\right)^4 - 6x^3 + 8x^2 + 49x + 10$
15. $F(x) = x\left(\arctan\frac{1}{(x+2)^2} + e^{-x^2-0.1}\right)$
16. $F(x) = x\,e^{-\frac{1}{(x+1)^2+1}} - (x+1)\,e^{-\frac{1}{(x-1)^2+1}}$
17. $F(x) = \left(e^{-x^2} + 0.03\arctan x\right)(x-0.5)$
18. $F(x) = \cos\left(\frac{1}{x^2+0.2}\right) + x$
19. $F(x) = \sin\left(\frac{1}{x^2+0.18}\right) + x$
20. $F(x) = \sin\left(\frac{1}{\ln(x^2+1.18)}\right) + x$
21. $F(x) = \sin\left(\frac{1}{x^2+0.4}\right) + \arctan(x+1)$

22. $F(x)=\cos\left(\frac{1}{(x+1)^2+0.6}\right)+\ln\left(\frac{1}{(x-1)^2+1.1}\right)$

23. $F(x)=\frac{\ln\left(x^2+2x+1.1\right)}{1.1-e^{-x^2}}$

24. $F(x)=\sin\left(\frac{1}{x^2+0.3}\right)+\ln\left((x+1)^2+1\right)$

25. $F(x)=\cos\left(\frac{1}{(x+1)^2+0.6}\right)+\sin\left(\frac{1}{(x-1)^2+1.5}\right)$

26. $F(x)=\cos\left(\frac{1}{(x+1)^2+0.6}\right)-\sin\left(\frac{1}{(x-1)^2+1}\right)$

Варианты заданий к теме “Многомерная оптимизация”

1. $F(x,y)=-\frac{1}{\sqrt{x^2+x+2}+\sqrt{y^2+\pi y+8}}\to\min$

2. $F(x,y)=\frac{1}{\sqrt{2x^2+x+y^2+3}+\sqrt{x^2+2y^2-3y+5}}\to\max$

3. $F(x,y)=-\frac{1}{x^2+xy+4y^2+2+2\sin(xy-x)}\to\min$

4. $F(x,y)=\frac{1}{x^2+xy+4y^2+2+2\cos(xy-y+1)}\to\max$

5. $F(x,y)=(x+2y)\arctan\frac{1}{x^2+3y^2+1}\to\min$

6. $F(x,y)=(x-y)\arctan\frac{1}{(x-1)^2+(y+1)^2+1}\to\max$

7. $F(x,y)=e^{-\frac{1}{x^2+y^2+\arctan(y+x/2+1)+2}}\to\min$

8. $F(x,y)=-e^{-\frac{1}{x^2+y^2+\arctan(-y+2x-1)+2}}\to\max$

9. $F(x,y)=e^{-\frac{x+2y}{x^2+y^2+1}}\to\min$

10. $F(x,y)=-e^{-\frac{x+1+2\cos y}{(x-1)^2+y^2+2}}\to\max$

11. $F(x,y)=x^4+y^4+5y^3+20x^2+15y^2+2\to\min$

12. $F(x,y)=3\arctan\left((x+2)^2+\dfrac{(y-2)^2}{3}-2xy^2\right)-x^2-\dfrac{y^2}{3}\ \to \max$

13. $F(x,y)=3\arctan(x-y+2)-x^2-\dfrac{y^2}{3}\ \to \min$

14. $F(x,y)=e^{-\frac{1}{x^2+y^2+2+\sin(x(y+1))}}\ \to \min$

15. $F(x,y)=-e^{-\frac{1}{x^2+y^2+2+\cos\left(0.3(x+1)(y-1)^2+1\right)}}\ \to \max$

16. $F(x,y)=e^{-\frac{1}{x^2+y^2+2+5\arctan\left((x-1)^2+(y+1)^2+1\right)}}\ \to \min$

17. $F(x,y)=e^{-\frac{1}{x^2+y^2+2+\cos\left(0.4\left((x+2)^2-(y-2)^2+2\right)\right)}}\ \to \min$

18. $F(x,y)=e^{-\frac{1}{x^2+y^2+2+\sin\left(0.3\left((x+2)^2-(y-2)^2+1\right)\right)}}\ \to \min$

19. $F(x,y)=e^{-x^2-y^2}(2x-y+1)\ \to \max$

20. $F(x,y)=-e^{-x^2-y^2}\left((x+2)^3-(y-2)^2+1\right)\ \to \min$

21. $F(x,y)=-e^{-x^2-y^2}\left((x-1)^3-(y+1)^3+1\right)\ \to \max$

22. $F(x,y)=\cos\dfrac{1}{(x+0.3)^2+(y-0.2)^2+0.4}+e^{-x^2-y^2+2}\ \to \max$

23. $F(x,y)=e^{-x^2-y^2+2}\cdot\cos\dfrac{1}{(x+0.3)^2+(y-0.2)^2+0.4}\ \to \min$

24. $F(x,y)=-e^{-x^2-y^2+2}\cdot\cos\left(0.5\left((x+1)^2+(y+1)^2+0.4\right)\right)\ \to \min$

25. $F(x,y)=-e^{-x^2-y^2+2}\cdot\sin\left(0.7\left((x+1)^2+(y+1)^2+0.8\right)\right)\ \to \min$

26. $F(x,y)=\sin\dfrac{1}{\ln\left(x^2+y^2+1.38\right)+\dfrac{0.8}{\ln\left((x-1)^2+(y+1)^2+1.38\right)}}\ \to \min$

Варианты заданий к теме “Системы нелинейных уравнений”

1. $\begin{cases}-\dfrac{x\sin(x+1)}{2}-y=1.2;\\ x+\dfrac{y\cos y}{2}=0.\end{cases}$

2. $\begin{cases}-\dfrac{x\sin(x+2)}{2}-y=1.2;\\ x+\dfrac{y\cos(y-1)}{2}=-2.5.\end{cases}$

3. $\begin{cases}-\arctan(y-1)+x=0;\\ -x\sin(x+2)-y=0.\end{cases}$

4. $\begin{cases}-\arctan(y-2)+x=0;\\ x\cos(x+2)-y=0.\end{cases}$

5. $\begin{cases} \arctan(y-1)+x=0; \\ -x\sin(x+2)-y=0. \end{cases}$

6. $\begin{cases} y\arctan(y-1)+x=2; \\ \sin(x+2)-y=0. \end{cases}$

7. $\begin{cases} -y\arctan(y-1)+x=-2; \\ \cos(x-1)-y=1. \end{cases}$

8. $\begin{cases} -e^{-y^2}+x=0; \\ -x\sin(x+2)-y=0. \end{cases}$

9. $\begin{cases} e^{-y^2}+x=0; \\ -\sin(x^2+2)-y=0. \end{cases}$

10. $\begin{cases} e^{-y^2+1}+x=1; \\ -\cos(x-1)-y=0. \end{cases}$

11. $\begin{cases} -\arctan(y-1)+x=0; \\ e^{\sin(x+2)}+y=0. \end{cases}$

12. $\begin{cases} -\arctan(y-1)+x=0; \\ e^{\sin(x+2)}-y=1. \end{cases}$

13. $\begin{cases} -\arctan(y+1)+x=0; \\ e^{\sin(x+2)}+y=0. \end{cases}$

14. $\begin{cases} -y\arctan(y+1)+x=-1; \\ e^{\cos(x+2)}+y=0. \end{cases}$

15. $\begin{cases} -\arctan(y+1)+x=0; \\ \sin(e^{-x^2}+2)+y=0. \end{cases}$

16. $\begin{cases} \arctan(y-2)+x=0; \\ \cos(e^{-x^2}+2)-y=-2. \end{cases}$

17. $\begin{cases} -\arctan(e^{-y^2}-1)+x=0; \\ \sin(x+2)-y=0. \end{cases}$

18. $\begin{cases} \arctan\left(\dfrac{y}{y^2+1}\right)+x=0; \\ \sin(x+2)-y=0. \end{cases}$

19. $\begin{cases} \arctan\left(\dfrac{y}{y^2+1}\right)+x=0; \\ (x+1)\sin(x+2)-y=0. \end{cases}$

20. $\begin{cases} 2\cos\left(\dfrac{y}{y^2+1}\right)+x=0; \\ \cos(x+2)-y=0. \end{cases}$

21. $\begin{cases} 2y(y+1)\sin\left(\dfrac{y}{y^2+1}\right)+x=0; \\ \arctan(x+1)-y=1. \end{cases}$

22. $\begin{cases} \dfrac{\sin 3y}{y}+x=0; \\ \cos(x+2)-y=0. \end{cases}$

23. $\begin{cases} \arctan(y^2+1)+x=0; \\ x\cos(x+2)-y=0. \end{cases}$

24. $\begin{cases} \arctan(y^2+1)+x=0; \\ \cos(x+2)-y=0. \end{cases}$

25. $\begin{cases} -\dfrac{\sin 3y}{2y}+x=-1; \\ \cos(x-1)-y=0. \end{cases}$

26. $\begin{cases} \dfrac{\sin 2y}{y}+x=0; \\ -\arctan(x-2)-y=0. \end{cases}$

Литература

1. *Абрамян М.Э., Олифер А.В*. Численные методы. Методические указания к выполнению индивидуальных заданий. – Ростов-на-Дону: УПЛ РГУ, 1991. 24 с.

2. *Михалкович С.С., Обрезанова О.А., Олифер А.В.* Численные методы. Методические указания к выполнению индивидуальных заданий. Выпуск 2. – Ростов-на-Дону: УПЛ РГУ, 1995. 34 с.

3. *Бахвалов Н.С., Жидков Н.П., Кобельков Г.М.* Численные методы. – М.: Наука, 1987. 600 с.

4. *Каханер Д., Моулер К., Нэш С.* Численные методы и программное обеспечение. – М.: Мир, 1998. 576 с.

5. *Воробьева Г.Н., Данилова А.Н.* Практикум по вычислительной математике. – М.: Высшая школа, 1990. 208 с.

6. *Говорухин В.Н., Цибулин В.М.*. Введение в Maple. –М.: Мир, 1997. 208с.