**4.1. Метод GlobalSearch**

Предполагается, что целевая функция и функции ограничений являются гладкими. Алгоритм, реализующий метод **GlobalSearch**, включает ряд последовательных шагов. На первом шаге запускается локальный решатель **fmincon** с заданной начальной точкой x0. Если получена сходимость, то запоминаются начальная и конечная точки и достигнутое значение целевой функции, используемое в функции счета (the score function). Функция счета определяется как сумма значения целевой функции и произведения множителя на сумму нарушений ограничений в данной точке. Если точка допустимая, то значение функции счета равно значению целевой функции. Начальное значение множителя равно 1000, но в процессе поиска оно обновляется. На втором шаге генерируется множество пробных точек **NumTrialPoints** – потенциально стартовых точек. Для этого используется алгоритм Гловера, подобный алгоритму случайного поиска (точки должны быть рассеяны, их компоненты лежат в диапазоне –e4+1, e4+1)). Далее выполняется стадия 1 (Stage 1): во всех пробных точках стадии 1 (их число задается параметром NumStageOnePoints как часть NumTrialPoints) вычисляется функция счета и выбирается лучшая точка, из которой запускается fmincon, а все остальные точки удаляются из общего списка пробных точек. Затем инициализируются области притяжения (бассейны), счетчики и порог. Значение порога локального решателя (localSolverThreshold) первоначально равно минимуму их двух значений целевой функции в точках решений, полученных из начальной точки x0 и из лучшей пробной точки. В качестве начальных бассейнов принимаются две сферические области с центрами в точках двух полученных fmincon решений с радиусами, равными расстояниям от начальной точки до точки решения. В алгоритме используются два типа счетчиков, считающих число последовательно исследованных пробных точек: 1) оказавшихся в бассейне (для каждого бассейна один такой счетчик); 2) у которых функция счета больше порога. Все счетчики первоначально равны 0.

Далее выполняется основной цикл алгоритма, в котором исследуются оставшиеся от первой стадии пробные точки, определяются стартовые и из них проводится локальный поиск. На стадии 2 (Stage 2) берется очередная пробная точка p. Если она не лежит в существующих бассейнах, ее функция счета лучше порога и, возможно, она удовлетворяет установленным границам и/или неравенствам, то из нее запускается fmincon (запускать ли из всех точек или только удовлетворяющих ограничениям, определяет параметр StartPointsToRun).

При этом счетчики бассейнов и порога обнуляются, а множество решений обновляется. Если установлена сходимость поиска из точки p, полученное решение xp сравнивается со всеми уже имеющимися решениями по расстоянию между точками и/или по разности значений целевых функций. Если они больше заданных величин, которые можно изменять параметрами TolX и/или TolFun, то в множество решений GlobalOptimSolution добавляется новый элемент, иначе точка p считается эквивалентом ближайшей точки и может ее заменить при лучшем показателе сходимости (exitflag). При сходимости поиска из точки p порог устанавливается в значение функции счета этой точки, а радиус бассейна с центром xp принимается равным максимуму из существующего радиуса и расстояния между p и xp. При невыполнении в точке p приведенных выше условий локальный решатель не запускается и выполняются следующие действия. Увеличивается значение счетчиков бассейнов, содержащих точку p, и сбрасываются в ноль счетчики остальных бассейнов. Для каждого бассейна, счетчик которого равен принятому значению MaxWaitCycle, уменьшается его радиус умножением на (1 – BasinRadiusFactor). Если счетчик порога равен MaxWaitCycle, порог увеличивается по следующей формуле: новый порог:=порог + PenaltyThresholdFactor\*(1 +abs(threshold)), а счетчик сбрасывается в 0. Все используемые здесь параметры приве- дены в прил. 3. По каждым 200 пробным точкам выдается одна строка отчета глобальных итераций.

Алгоритм завершает работу по истечении установленного значения времени или исчерпании пробных точек. Результатом является множество решений GlobalOptimSolution, упорядоченное по значениям целевой функции. Очевидно, что реализованный в алгоритме подход является эвристическим.

Теперь остановимся на применении GlobalSearch. Сначала создается структура Задача (problem) с помощью функции createOptimProblem, для чего задаются целевая функция в виде m-файла или анонимно, границы и функции нелинейных ограничений (при их наличии), стартовая точка xstart, например, для трехмерной задачи так:

**xstart = randn(3,1).**

Кроме того, если требуется задать параметры options, отличающиеся от установленных по умолчанию значений, то используется optimset (например, чтобы задать конкретный алгоритм fmincon). В качестве аргументов функции createOptimProblem сначала записывается имя решателя, затем параметры в виде пар «имя, значение». Для указания ограничений используются имена, приведенные ниже.

|  |  |
| --- | --- |
| **Ограничение** | **Имя** |
| Нижняя граница | 'lb' |
| Верхняя граница | 'ub' |
| Матрица линейных неравенств | 'Aineq' |
| Вектор ограничения линейных неравенств | 'bineq' |
| Матрица линейных равенств | 'Aeq' |
| Вектор ограничения линейных равенств | 'beq' |
| Функция нелинейных ограничений | 'nonlcon' |

Примечание. Поскольку в GlobalSearch используется только fmincon, то при отсутствии ограничений следует добавить искусственное:

**problem.lb = -Inf;**

Приведем вариант создания задачи. Например, зададим конкретный алгоритм fmincon

**opts = optimset('Algorithm','interior-point');**

определим целевую функцию с именем glmin и линейные неравенства с матрицей A и вектором b. Тогда для создания Задачи следует ввести

**problem =createOptimProblem('fmincon','x0',xstart,...**

**'objective',@glmin,'Aineq',A,'bineq',b,'options',opts);**

Теперь можно удостовериться в правильности созданной задачи запуском решателя fmincon:

**[x fval eflag output] = fmincon(problem);**

Для применения GlobalSearch принято создавать объект решателя (solver object), используемый в команде запуска run. Если параметры глобального поиска, установленные по умолчанию, приемлемы, то объект решателя с именем gs создается командой

**gs = GlobalSearch;**

Если же требуется изменить некоторые параметры, тогда они указываются как аргументы GlobalSearch в виде пар «имя, значение», например

**gs = GlobalSearch('TolX',0.01,'MaxTime',2000);,**

где заданы значения для выявления идентичных решений (когда различия между ними в пределах 0,01) и максимально допустимое время решения задачи. Все параметры, присущие GlobalSearch, приведены в прил. 3. Параметры локального решателя можно изменять, как и раньше, через optimset.

Запускается решение задачи командой run с аргументами gs и problem:

**[x,fval,exitflag,output,manyminsg] = run(gs,problem)**

Как и в обращениях к функциям, возвращаемых глобальных аргументов здесь может быть задано меньше, например [x,fval]. Выход exitflag дает признак причины завершения глобального решателя:

**2 –** найден по крайней мере один локальный минимум и некоторые запуски локального решателя привели к сходимости,

**1 –** найден по крайней мере один локальный минимум и все запуски локального решателя привели к сходимости,

**0 –** локальный минимум не найден, но локальный решатель вызывался не менее одного раза и превысил максимальное число итераций или оценок функции,

**-1 –** запущенный локальный решатель остановлен функцией output или plot,

**-2 –** не найден допустимый локальный минимум,

**-5 –** исчерпан лимит времени MaxTime,

**-8 –** решение не найдено, все запуски локального решателя завершились с exitflag <= -2,

**-10 –** ошибки в функциях, определенных пользователем.

Глобальная структура output содержит количество вычислений целевой функции, общее число запусков локального решателя (fmincon), из них успешных и неуспешных. Четвертый выходной аргумент Manyminsg представляет собой структуру глобального решения, которая включает массив X точек найденных минимумов, вектор Fval значений целевой функции в этих точках, вектор значений Exitflag локального решателя, структуру output и массив X0 стартовых точек. Очевидно, что эту структуру следует указывать в выходных аргументах в тех случаях, когда требуется получить данные обо всех найденных минимумах.

**Пример 7.**

**Найти глобальный минимум функции**

**f = 3(1– x1)2exp(– 2 1 x – (x2 + 1)2) –10( x1/5 – 3 1 x – 5 2 x )exp (– 2 1 x – 2 1 x ) – – 1/3exp (– (x1 + 1) – 2 2 x )**

Напомним, что данная функция представлена в m-файле с именем myfun.

Попытаемся найти глобальное решение методом GlobalSearch. Для этого создадим объект gs с параметром 'Display' в значении 'iter', что обеспечит вывод итераций главного цикла (Stage 2) глобального поиска. Затем выберем алгоритм локального поиска 'interior-point', создадим Задачу (problem) с целевой функцией myfun и обратимся к функции run. В итоге имеем следующую программу:

**gs = GlobalSearch('Display','iter');**

**opts = optimset('Algorithm','interior-point');**

**problem = createOptimProblem('fmincon','x0',x0,...**

**'objective',@myfun,'lb',[-3,-3],'ub',[3,3],...**

**'options',opts);**

**[xminm,fminm,flagm,outptm,manyminsg] = ...**

**run(gs,problem);**

Ее достаточно для поиска решения, но, чтобы придать наглядность решению, дополним ее трехмерным графиком функции, на котором в горизонтальной плоскости переменных отобразим линии уровня (контуры) функции, стартовые точки и центры бассейнов (найденные минимумы). При этом графические построения будем производить по функции myfun1, записанной в одноименном m-файле:

**function f=myfun1(x,y)**

**f=3\*(1-x).^2.\*exp(-x.^2-(y+1).^2)-10\*(x/5....**

**-x.^3-y.^5).\*exp(-x.^2-y.^2)-1/3\*exp(-(x...**

**+1).^2-y.^2);**

**end**

Для удобства многократных запусков оформим расширенную программу в виде функции от начальной точки с именем runglobal:

**function runglobal(x0)**

**[X,Y]=meshgrid(-4:0.1:4); Z=myfun1(X,Y);meshc(X,Y,Z);**

**xlabel('x1');ylabel('x2');zlabel('f');hold on;**

**gs = GlobalSearch('Display','iter');**

**opts = optimset('Algorithm','interior-point');**

**problem = createOptimProblem('fmincon','x0',x0,...**

**'objective',@myfun,'lb',[-3,-3],'ub',[3,3],...**

**'options',opts);**

**[xminm,fminm,flagm,outptm,manyminsg] = run(gs,problem)**

**% Plot points**

**possColors = 'kbgcrm';**

**hold on**

**for i = 1:size(manyminsg,2)**

**% Color of this line**

**cIdx = rem(i-1, length(possColors)) + 1;**

**color = possColors(cIdx);**

**% Plot start points**

**u = manyminsg(i).X0;**

**x0ThisMin = reshape([u{:}], 2, length(u));**

**z=-10\*ones(size(x0ThisMin));**

**plot3(x0ThisMin(1, :), x0ThisMin(2, :),z,'.', ...**

**'Color',color,'MarkerSize',12);**

**% Plot the basin with color i**

**z=-10\*ones(size(manyminsg(i).X));**

**plot3(manyminsg(i).X(1), manyminsg(i).X(2),z,'\*', ...**

**'Color', color, 'MarkerSize',10);**

**end**

**% basin center marked with a \*, start points**

**% with dots**

**end**

Чтобы сравнить с результатами применения локального метода в примере 6, запустим программу runglobal с начальной точки, которая тогда оказалась неудачной:

>> x0=[-0.5 3]; runglobal(x0)

После завершения вычислений выводится информация о больших итерациях в виде следующей таблицы:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Num Pts  Analyzed | F-count | Best f(x) | Current Penalty | Threshold Penalty | Local f(x) | Local exitflag | Procedure |
| 0 | 84 | 3.287e-005 |  |  | 3.287e-005 | 1 | Initial Point |
| 200 | 1321 | -3.05 |  |  | -3.05 | 1 | Stage 1 Local |
| 217 | 1383 | -6.551 | -3.461 | -3.05 | -6.551 | 1 | Stage 2 Local |
| 227 | 1429 | -6.551 | -4.196 | -3.461 | -6.551 | 1 | Stage 2 Local |
| 300 | 1502 | -6.551 | -0.06148 | -3.327 |  |  | Stage 2 Search |
| 400 | 1602 | -6.551 | -1.315e-005 | -0.4178 |  |  | Stage 2 Search |
| 422 | 1658 | -6.551 | -0.2512 | -0.1343 | -6.551 | 1 | Stage 2 Local |
| 491 | 1761 | -6.551 | -1.827 | -0.6872 | -3.05 | 1 | Stage 2 Local |
| 500 | 1770 | -6.551 | 0.04837 | -2.211 |  |  | Stage 2 Search |
| 529 | 1834 | -6.551 | -2.118 | -2.104 | -3.05 | 1 | Stage 2 Local |
| 600 | 1905 | -6.551 | -0.9791 | -1.729 |  |  | Stage 2 Search |
| 700 | 2005 | -6.551 | -1.429 | -6.25 |  |  | Stage 2 Search |
| 800 | 2105 | -6.551 | 0.009279 | -2.641 |  |  | Stage 2 Search |
| 879 | 2216 | -6.551 | -1.405 | -1.305 | -3.05 | 1 | Stage 2 Local |
| 880 | 2249 | -6.551 | -1.48 | -1.405 | -3.05 | 1 | Stage 2 Local |
| 900 | 2269 | -6.551 | -2.18 | -2.956 |  |  | Stage 2 Search |
| 1000 | 2369 | -6.551 | -6.318 | -6.525 |  |  | Stage 2 Search |

Здесь по каждой итерации представлены число проверенных пробных точек, количество вычислений целевой функции и лучшее из ее достигнутых значение, текущий и пороговые штрафы, минимум f(x) локального решателя и признак его завершения и стадия алгоритма.

Далее выводится сообщение:

**GlobalSearch stopped because it analyzed all the trial points.**

**All 9 local solver runs converged with a positive local solver exit flag.**

Оно информирует, что алгоритм остановился, так как проанализированы все пробные точки, все девять запусков локального решателя завершились с положительным признаком (в таблице значение Exitflag=1 соответствует всем девяти локальным поискам).

Следующая порция вывода содержит основные итоговые результаты:

– координаты лучшей точки

xminm = 0.2283 -1.6255

– значение целевой функции в этой точке

fminm = -6.5511

– признак завершения лучшего поиска

flagm = 1

– структуру вывода, в которой дается общее количество обращений к целевой функции, общее число запусков локального решателя и из них удачных и неудачных

outptm =

funcCount: 2369

localSolverTotal: 9

localSolverSuccess: 9

localSolverIncomplete: 0

localSolverNoSolution: 0

message: [1x137 char]

Завершается вывод в командное окно представлением глобальной структуры

manyminsg =

1x3 GlobalOptimSolution

Properties:

X

Fval

Exitflag

Output

X0

Наконец, в графическом окне мы видим целевую функцию и точки, приводящие к минимумам (рис. 11).



Рис. 11. Поиск глобального минимума методом GlobalSearch

На этом рисунке ромбиками показаны стартовые точки, приведшие к минимуму, а снежинками – центры бассейнов притяжения (локальные минимумы). Одному минимуму соответствует один цвет ромбиков и снежинок. Интересно, что один из обнаруженных минимумов (зеленая снежинка) соответствует тому, в котором «застрял» локальный решатель в примере 6. Если нас интересуют все минимумы, то получить их координаты и соответствующие значения целевой функции можно из структуры manyminsg (см. программу runglobal).

Как следует из полученных численных и графических данных, в процессе поиска по методу GlobalSearch исследовано 1000 пробных точек, из них восемь точек использованы как стартовые для локального решателя, при этом из пяти точек достигнут локальный минимум со значением целевой функции –3,05 (эти точки имеют синий цвет), из трех точек – локальный минимум со значением –6,551 (черный цвет), а начальная точка x0 привела в минимум со значением 3.287e-005 (зеленый цвет). В целом поиск оказался успешным: в точке (0,2283 –1,6255) найден глобальный минимум со значением целевой функции –6,5511.

**4.2. Метод MultiStart**

К целевой функции и функциям ограничений решаемой задачи предъявляются такие же требования, как в GlobalSearch. Методы GlobalSearch и MultiStart базируются на общей идее: поиск минимумов осуществляется запуском локального решателя из множества стартовых точек, а отличаются они способами формирования и выбора этих точек. Кроме того, в MultiStart можно использовать не только fmincon, но и другие локальные решатели, а также распараллеливать процесс.

Объект MultiStart создается выражением

**ms = MultiStart,**

где ms – имя объекта, его глобальные свойства (параметры) приведены в прил. 3. Задача (problem) создается так же, как в GlobalSearch. Выполнение алгоритма осуществляется функцией run.

Основными шагами алгоритма MultiStart являются генерация стартовых точек, фильтрация сгенерированных точек и запуск локального решателя. Подробнее остановимся на первом шаге.

Имеется четыре способа задания набора стартовых точек.

1. Использовать объект RandomStartPointSet, генерирующий псевдослучайные числа. Он создается предложением

**stpoints = RandomStartPointSet;**

По умолчанию такой объект генерирует 10 стартовых точек. Если требуется другое число точек, оно указывается как значение параметра NumStartPoints, например

**stpoints = RandomStartPointSet('NumStartPoints',70)**

задает 70 точек. На множество генерируемых точек можно накладывать искусственные границы через параметр ArtificialBound

**stpoints=RandomStartPointSet('NumStartPoints',70, ...**

**'ArtificialBound',50).**

Если задача не содержит границ, то нижней границей будет –AB, а верхней AB, где AB – значение ArtificialBound. При наличии только нижней границы lb верхняя граница будет равна lb+2\*AB. А когда есть только верхняя граница ub, нижней границей будет ub-2\*AB.

2. Задать число стартовых точек S как входной параметр функции run. В этом случае будет генерироваться S-1 псевдослучайных точек, к которым добавляется начальная точка x0.

3. Создать пользовательский объект CustomStartPointSet. Для

этого сначала формируют матрицу, в которой каждая строка – это координаты стартовой точки. Например, матрицу pts из 50 точек четырехмерного пространства, равных 20 плюс экспоненциально распределенное случайное число со средним значением 10, можно получить так:

**pts = -10\*log(rand(50,4)) + 20;**

Затем на этой матрице создается объект:

**tpoints = CustomStartPointSet(pts).**

Чтобы увидеть матрицу pts, можно воспользоваться функцией list:

**pts = list(tpoints).**

4. Использовать несколько наборов точек. Например, для образования множества стартовых точек с использованием приведенных выше объектов CustomStartPointSet и RandomStartPointSet следует ввести

**pts = -10\*log(rand(50,4)) + 20;**

**tpoints = CustomStartPointSet(pts);**

**rpts = RandomStartPointSet('NumStartPoints',40);**

**allpts = {tpoints,rpts}.**

Включение allpts в качестве параметра функции run обеспечит генерирование 90 стартовых точек.

Теперь рассмотрим второй шаг алгоритма. Фильтрация сгенерированных точек имеет место, когда параметр StartPointsToRun равен 'bounds' или 'bounds-ineqs': в качестве стартовых будут использованы только точки, удовлетворяющие указанным ограничениям. По умолчанию используются все точки (параметр = 'all').

На третьем шаге MultiStart запускает локальный решатель, указанный в задаче (problem). В качестве него может быть выбран один из следующих решателей: fmincon, fminunc, lsqcurvefit, lsqnonlin. Последние два решателя предназначены для задач подбора кривой и неотрицательного линейного метода наименьших квадратов соответственно (см. гл. 9). Если установлен параллельный режим, то стартовые точки распределяются по работающим процессорам, которые запускают локальный решатель. Если решатель остановился из-за исчерпания лимита времени, то решение не фиксируется, иначе запоминаются результаты и выполняется следующий шаг. Алгоритм завершается, когда использованы все стартовые точки или время решения превысило допустимое (MaxTime). При этом, как и в GlobalSearch, формируется множество решений GlobalOptimSolution, упорядоченное по значениям целевой функции. Из каждого множества идентичных решений, которые выявляются с использование параметров TolFun и TolX, в него попадает только одно.

Далее рассмотрим на примерах применение метода MultiStart.

Пример 8.

Найти минимумы функции



при условиях



Целевую функцию представим в виде m-файла с именем sixmin:

**function f = sixmin(x)**

**f=4\*x(1)^2 - 2.1\*x(1)^4 + x(1)^6/3 ...**

**+ x(1)\*x(2) - 4\*x(2)^2 + 4\*x(2)^4;**

**end**

Создаем объект со свойством вывода информации по итерациям:

**ms = MultiStart('Display','iter');**

в качестве локального решателя выбираем fmincon с алгоритмом внутренней точки и создаем проблему:

**opts = optimset('Algorithm','interior-point');**

**problem = createOptimProblem('fmincon','x0',x0,...**

**'objective',@sixmin,'lb',[-3,-3],'ub',[3,3],...**

**'options',opts);**

и, наконец, задаем 40 стартовых точек в команде запуска run:

**[x,fval] = run(ms,problem,40)**

Такой программы достаточно для решения нашей задачи. Но чтобы наглядно продемонстрировать работу метода, расширим программу, добавив команды вывода 3D-графика функции, ее линий уровня и отображения стартовых и конечных точек локальных поисков. Для отображения графики создадим второй m-файл целевой функции:

**function fg = sixmin\_g(x,y)**

**fg=4\*x.^2 - 2.1\*x.^4 + x.^6/3 ...**

**+ x.\*y - 4\*y.^2 + 4\*y.^4;**

**end**

Кроме того, расширим вывод функции run, и в итоге получаем следующую программу, оформленную как функция runglobal1:

**function runglobal1(x0)**

**[x,y]=meshgrid(-3:0.03:3);Z=sixmin\_g(x,y);**

**meshc(x,y,Z);**

**xlabel('x1');ylabel('x2');zlabel('f');**

**figure; c=contour(x,y,Z,[-0.8 -0.2 0.07 1 3 7 ...**

**15 28 45 65],'blue');**

**xlabel('x1');ylabel('x2');**

**ms = MultiStart('Display','iter');**

**opts = optimset('Algorithm','interior-point');**

**problem = createOptimProblem('fmincon','x0',x0,...**

**'objective',@sixmin,'lb',[-3,-3],'ub',[3,3],...**

**'options',opts);**

**[x,fval,flag,outpt,manyminms] = run(ms,problem,40)**

**% Plot points**

**possColors = 'kbgcrm';**

**hold on**

**for i = 1:size(manyminms,2)**

**% Color of this line**

**cIdx = rem(i-1, length(possColors)) + 1;**

**color = possColors(cIdx);**

**% Plot start points**

**u = manyminms(i).X0;**

**x0ThisMin = reshape([u{:}], 2, length(u));**

**plot(x0ThisMin(1, :), x0ThisMin(2, :),'.', ...**

**'Color',color,'MarkerSize',12);**

**% Plot the basin with color i**

**plot(manyminms(i).X(1), manyminms(i).X(2),'\*', ...**

**'Color', color, 'MarkerSize',10);**

**end % basin center marked with a \*, start points**

**% with dots**

**end**

С произвольно взятой начальной точки запускаем программу

**>> x0=[0 1]; runglobal1(x0)**

Сначала выводится график функции с контурами на горизонтальной плоскости (рис. 12), на котором визуально обнаруживается несколько минимумов. Затем следует отчет по итерациям, строки которого соответствуют всем стартовым точкам, включая начальную:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Run Index | Local exitflag | Local f(x) | Local # iter | Local F-count | First-order optimality |
| 1 | 1 | -1.032 | 9 | 35 | 8.744e-007 |
| 2 | 1 | -1.032 | 17 | 55 | 8.745e-007 |
| 3 | 1 | -1.032 | 8 | 31 | 8.745e-007 |
| 4 | 1 | -1.032 | 8 | 31 | 8.741e-007 |
| 5 | 1 | -1.032 | 14 | 47 | 1.749e-007 |
| 6 | 1 | -1.032 | 8 | 30 | 8.744e-007 |
| 7 | 1 | -1.032 | 10 | 35 | 8.745e-007 |
| 8 | 1 | -0.2155 | 11 | 58 | 3.742e-008 |
| 9 | 1 | -1.032 | 10 | 37 | 8.744e-007 |
| 10 | 1 | -1.032 | 11 | 40 | 8.745e-007 |
| 11 | 1 | -1.032 | 15 | 51 | 2.645e-008 |
| 12 | 1 | -1.032 | 10 | 35 | 8.743e-007 |
| 13 | 1 | -1.032 | 11 | 40 | 8.745e-007 |
| 14 | 1 | -1.032 | 10 | 37 | 1.154e-008 |
| 15 | 1 | -1.032 | 8 | 32 | 8.745e-007 |
| 16 | 1 | -1.032 | 13 | 44 | 1.749e-007 |
| 17 | 1 | -0.2155 | 12 | 44 | 4.616e-008 |
| 18 | 1 | -1.032 | 11 | 41 | 8.838e-007 |
| 19 | 1 | -1.032 | 13 | 48 | 8.744e-007 |
| 20 | 1 | -0.2155 | 11 | 44 | 2.381e-008 |
| 21 | 1 | -1.032 | 11 | 40 | 8.743e-007 |
| 22 | 1 | -0.2155 | 9 | 34 | 9.864e-008 |
| 23 | 1 | -1.032 | 12 | 43 | 9.457e-007 |
| 24 | 1 | -1.032 | 11 | 38 | 4.135e-008 |
| 25 | 1 | -1.032 | 13 | 48 | 8.744e-007 |
| 26 | 1 | -1.032 | 9 | 33 | 8.743e-007 |
| 27 | 1 | -0.2155 | 13 | 47 | 2.381e-008 |
| 28 | 1 | -0.2155 | 15 | 53 | 9.864e-008 |
| 29 | 1 | -1.032 | 8 | 32 | 8.743e-007 |
| 30 | 1 | -0.2155 | 12 | 43 | 1.074e-007 |
| 31 | 1 | -0.2155 | 11 | 41 | 4.09e-008 |
| 32 | 1 | 2.104 | 13 | 45 | 2.872e-007 |
| 33 | 1 | -1.032 | 15 | 60 | 8.745e-007 |
| 34 | 1 | -0.2155 | 13 | 45 | 1.025e-007 |
| 35 | 1 | -1.032 | 10 | 36 | 8.744e-007 |
| 36 | 1 | -1.032 | 9 | 33 | 8.745e-007 |
| 37 | 1 | -1.032 | 15 | 53 | 8.745e-007 |
| 38 | 1 | -0.2155 | 9 | 39 | 3.037e-007 |
| 39 | 1 | -1.032 | 9 | 32 | 8.745e-007 |
| 40 | 1 | -1.032 | 9 | 32 | 8.743e-007 |

MultiStart completed the runs from all start points.

All 40 local solver runs converged with a positive local solver exit flag.

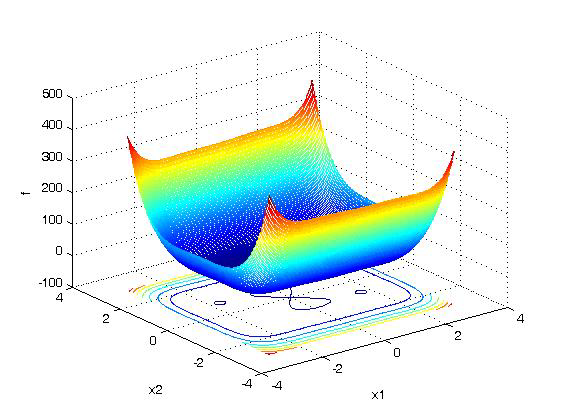


Рис. 12. Функция sixmin

Из этого отчета следует, что поиск из всех 40 точек завершился успешно (exitflag везде равен 1), при этом получены три минимальных значения целевой функции.

Затем выводятся данные по лучшему минимуму и структуры outpt и manyminms:

**x =**

**-0.0898 0.7127**

**fval = -1.0316**

**flag = 1**

**outpt =**

**funcCount: 1645**

**localSolverTotal: 40**

**localSolverSuccess: 40**

**localSolverIncomplete: 0**

**localSolverNoSolution: 0**

**message: [1x128 char]**

**manyminms =**

**1x5 GlobalOptimSolution**

**Properties:**

**X**

**Fval**

**Exitflag**

**Output**

**X0**

Из последней части отчета имеем: самое минимальное значение функции цели –1,0316 достигнуто в точке (–0,0898 0,7127), при этом функция вычислялась 1645 раз и обнаружено пять локальных минимумов.

Завершает вывод карта линий уровня (контуров) целевой функции с нанесенными на нее стартовыми точками (цветные точки) и точками найденных минимумов (обозначены символом \*). Она представлена на рис. 13.

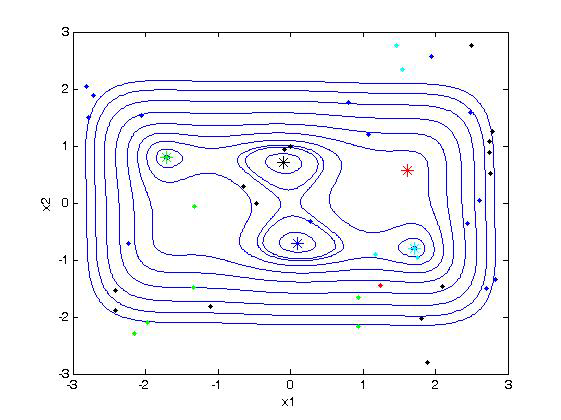


Рис. 13. Линии уровня функции sixmin и точки начала

и окончания локальных поисков

Рис. 13 подтверждает обнаружение пяти минимумов, из них две пары минимумов оказались симметричными относительно начала координат, и поэтому минимальных значений функции только три.

Запустим программу повторно из той же начальной точки, убрав предварительно вывод отчета по итерациям. В результате получаем

**x =**

**0.0898 -0.7127**

**fval = -1.0316**

**flag = 1**

**outpt =**

**funcCount: 1589**

**localSolverTotal: 40**

**localSolverSuccess: 40**

**localSolverIncomplete: 0**

**localSolverNoSolution: 0**

**message: [1x128 char]**

**manyminms =**

**1x6 GlobalOptimSolution**

**Properties:**

**X**

**Fval**

**Exitflag**

**Output**

**X0**

Расположение стартовых точек и найденных минимумов показано на рис. 14.

Мы видим, что теперь при несколько меньшем числе вычислений целевой функции обнаружено шесть минимумов, из них три пары симметрично расположенных относительно начала координат. При многократных повторах выполнения программы обнаруживается от четырех до шести минимумов, и всегда в их число попадают два симметричных минимума с , и , которые и являются глобальными.

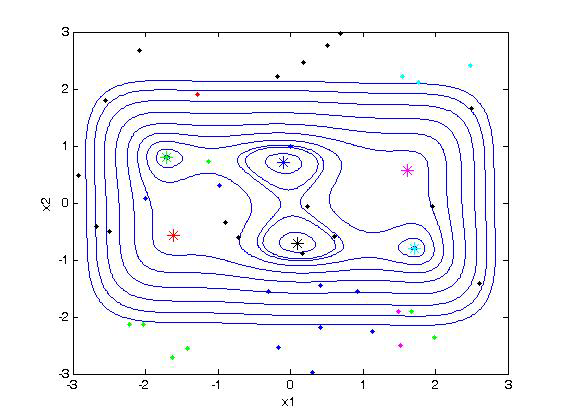
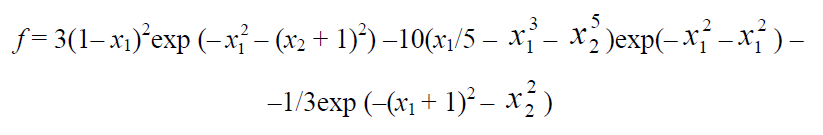


Рис. 14. Результаты повторного поиска минимумов

функции sixmin

Пример 9.

Методом MultiStart найти минимумы уже исследованной ранее функции



при условиях



Для решения этой задачи составим программу, подобную runglobal1, но отличающуюся тем, что она позволит на одном графике отображать и функцию, и контуры, и все точки, фигурирующие в процессе поиска:

**function runglobal1\_2(x0)**

**[X,Y]=meshgrid(-4:0.1:4);**

**Z=myfun1(X,Y);meshc(X,Y,Z);**

**xlabel('x1');ylabel('x2');zlabel('f');**

**ms = MultiStart('Display','iter');**

**opts = optimset('Algorithm','interior-point');**

**problem = createOptimProblem('fmincon','x0',x0,...**

**'objective',@myfun,'lb',[-3,-3],'ub',[3,3],...**

**'options',opts);**

**[xm,fm,flagm,outptm,manyminms] = run(ms,problem,40)**

**possColors = 'kbgcrm';**

**hold on**

**for i = 1:size(manyminms,2)**

**% Color of this line**

**cIdx = rem(i-1, length(possColors)) + 1;**

**color = possColors(cIdx);**

**% Plot start points**

**u = manyminms(i).X0;**

**x0ThisMin = reshape([u{:}], 2, length(u));**

**z=-10\*ones(size(x0ThisMin));**

**plot3(x0ThisMin(1, :), x0ThisMin(2, :),z, '.', ...**

**'Color',color,'MarkerSize',12);**

**% Plot the basin with color i**

**z=-10\*ones(size(manyminms(i).X));**

**plot3(manyminms(i).X(1),manyminms(i).X(2),z,'\*', ...**

**'Color', color, 'MarkerSize',10);**

**end % basin center marked with a \*, start points**

**% with dots**

**end**

Задаем начальную точку и запускаем нашу программу:

>> x0=[3 -0.5];runglobal1\_2(x0)

Через несколько секунд появляется отчет по итерациям:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Run Index | Local exitflag | Local f(x) | Local # iter | Local F-count | First-order optimality |
| 1 | 1 | -6.551 | 17 | 63 | 6.311e-008 |
| 2 | 1 | -6.551 | 10 | 39 | 2.394e-007 |
| 3 | 1 | -6.551 | 10 | 38 | 2.394e-007 |
| 4 | 1 | 6.687e-005 | 20 | 63 | 5.184e-007 |
| 5 | 1 | -3.05 | 10 | 37 | 1.21e-008 |
| 6 | 1 | -6.551 | 14 | 55 | 2.374e-007 |
| 7 | 1 | -3.05 | 29 | 99 | 1.406e-007 |
| 8 | 1 | -6.551 | 15 | 54 | 2.99e-007 |
| 9 | 1 | 6.686e-005 | 22 | 69 | 7.893e-007 |
| 10 | 1 | -6.551 | 22 | 110 | 1.182e-007 |
| 11 | 1 | 4.106e-005 | 26 | 81 | 5.769e-008 |
| 12 | 1 | -6.551 | 10 | 38 | 1.202e-007 |
| 13 | 1 | 4.125e-005 | 24 | 75 | 8.355e-007 |
| 14 | 1 | -3.05 | 11 | 43 | 5.885e-008 |
| 15 | 1 | -6.551 | 10 | 43 | 2.394e-007 |
| 16 | 1 | -6.551 | 9 | 37 | 6.063e-008 |
| 17 | 1 | -6.551 | 12 | 48 | 5.858e-008 |
| 18 | 1 | -6.551 | 10 | 41 | 9.978e-008 |
| 19 | 1 | -3.05 | 11 | 39 | 1.462e-008 |
| 20 | 1 | -3.05 | 11 | 44 | 1.481e-007 |
| 21 | 1 | -3.05 | 12 | 43 | 3.072e-008 |
| 22 | 1 | 6.687e-005 | 20 | 63 | 5.012e-007 |
| 23 | 1 | -6.551 | 9 | 42 | 1.569e-007 |
| 24 | 1 | -3.05 | 16 | 53 | 9.032e-008 |
| 25 | 1 | 3.227e-005 | 26 | 81 | 6.415e-008 |
| 26 | 1 | -3.05 | 8 | 33 | 5.869e-008 |
| 27 | 1 | -6.551 | 12 | 45 | 1.798e-007 |
| 28 | 1 | 4.125e-005 | 27 | 84 | 6.959e-007 |
| 29 | 1 | -3.05 | 15 | 52 | 5.174e-008 |
| 30 | 1 | -6.551 | 21 | 71 | 1.182e-007 |
| 31 | 1 | -6.551 | 16 | 54 | 9.978e-008 |
| 32 | 1 | -6.551 | 12 | 50 | 2.394e-007 |
| 33 | 1 | -3.05 | 12 | 43 | 8.849e-008 |
| 34 | 1 | -6.551 | 13 | 58 | 8.356e-008 |
| 35 | 1 | -3.05 | 9 | 34 | 1.21e-008 |
| 36 | 1 | -3.05 | 11 | 42 | 3.072e-008 |
| 37 | 1 | -3.05 | 12 | 42 | 9.032e-008 |
| 38 | 1 | -6.551 | 10 | 39 | 2.394e-007 |
| 39 | 1 | -3.05 | 13 | 45 | 1.183e-007 |
| 40 | 1 | -6.551 | 21 | 72 | 6.311e-008 |

MultiStart completed the runs from all start points.

All 40 local solver runs converged with a positive local solver exit flag.

После таблицы выдано сообщение, в котором констатируется, что метод завершил работу по причине использования для поиска всех стартовых точек, и все старты локального решателя привели к сходимости. Последующая часть текстового вывода содержит итоги работы MultiStart:

– точку предполагаемого глобального минимума и его значение

**xm =**

**0.2283 -1.6255**

**fm =**

**-6.5511**

**– признак завершения глобального поиска и его структуры**

**flagm = 1**

**outptm =**

**funcCount: 2165**

**localSolverTotal: 40**

**localSolverSuccess: 40**

**localSolverIncomplete: 0**

**localSolverNoSolution: 0**

**message: [1x128 char]**

**manyminms =**

**1x8 GlobalOptimSolution**

**Properties:**

**X**

**Fval**

**Exitflag**

**Output**

**X0**

Таким образом, поиски из 40 стартовых точек позволили выявить восемь точек, в которых возможен глобальный минимум. Как легли эти точки на заданную область поиска, показано на рис. 15. Там же символом \* отмечены пять локальных минимумов, включая глобальный минимум в точке (0,2283 –1,6255) со значением целевой функции –6,5511. Это же решение было получено методом GlobalSearch (пример 7), которому потребовалось вычислять целевую функцию 2369 раз против 2165 методом MultiStart.

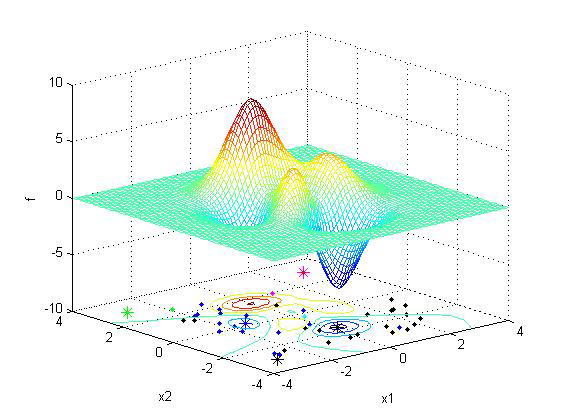


Рис. 15. Поиск минимумов функции myfun методом MultiStart

Пример 10.

Найти максимумы функции из примера 9 при тех же условиях.

Чтобы воспользоваться тем же методом, изменим знак функции и запишем ее в m-файл с именем myfun2, заменим ею функцию myfun в runglobal1\_2 и новую версию программы запишем под именем runglobal1\_3.

Результаты, полученные при выполнении этой программы, приведены ниже.

**>> x0=[3 -0.5]; runglobal1\_3(x0)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Run Index | Local exitflag | Local f(x) | Local # iter | Local F-count | First-order optimality |
| 1 | 1 | -3.592 | 11 | 39 | 5.336e-008 |
| 2 | 1 | -3.592 | 10 | 36 | 6.254e-008 |
| 3 | 1 | -8.106 | 9 | 36 | 6.565e-008 |
| 4 | 1 | -3.777 | 7 | 37 | 8.437e-007 |
| 5 | 1 | -3.592 | 19 | 69 | 3.018e-008 |
| 6 | 1 | -3.592 | 21 | 77 | 2.333e-007 |
| 7 | 1 | -8.106 | 11 | 40 | 1.193e-007 |
| 8 | 1 | -8.106 | 22 | 76 | 2.82e-007 |
| 9 | 1 | -8.106 | 13 | 49 | 1.193e-007 |
| 10 | 1 | -8.106 | 24 | 112 | 1.41e-007 |
| 11 | 1 | -3.592 | 10 | 36 | 5.336e-008 |
| 12 | 1 | -8.106 | 10 | 38 | 1.193e-007 |
| 13 | 1 | -3.777 | 9 | 35 | 8.437e-007 |
| 14 | 1 | -3.592 | 23 | 84 | 5.958e-008 |
| 15 | 1 | -8.106 | 10 | 37 | 1.605e-007 |
| 16 | 1 | -8.106 | 30 | 104 | 1.194e-007 |
| 17 | 1 | -8.106 | 20 | 68 | 1.193e-007 |
| 18 | 1 | -8.106 | 15 | 76 | 2.385e-007 |
| 19 | 1 | -8.106 | 31 | 101 | 5.952e-007 |
| 20 | 1 | -8.106 | 31 | 104 | 8.512e-008 |
| 21 | 1 | -3.777 | 10 | 38 | 8.437e-007 |
| 22 | 1 | -8.106 | 21 | 81 | 2.82e-007 |
| 23 | 1 | -3.777 | 14 | 50 | 8.731e-008 |
| 24 | 1 | -8.106 | 11 | 42 | 5.96e-007 |
| 25 | 1 | -8.106 | 18 | 61 | 1.192e-007 |
| 26 | 1 | -8.106 | 17 | 61 | 2.82e-007 |
| 27 | 1 | -8.106 | 9 | 37 | 3.577e-007 |
| 28 | 1 | -3.777 | 9 | 36 | 5.668e-008 |
| 29 | 1 | -8.106 | 10 | 38 | 2.385e-007 |
| 30 | 1 | -8.106 | 10 | 37 | 2.385e-007 |
| 31 | 1 | -3.592 | 12 | 40 | 1.192e-007 |
| 32 | 1 | -3.777 | 9 | 34 | 8.435e-007 |
| 33 | 1 | -8.106 | 18 | 63 | 8.512e-008 |
| 34 | 1 | -8.106 | 11 | 42 | 3.576e-007 |
| 35 | 1 | -8.106 | 15 | 52 | 8.512e-008 |
| 36 | 1 | -8.106 | 12 | 44 | 1.41e-007 |
| 37 | 1 | -3.777 | 15 | 54 | 8.437e-007 |
| 38 | 1 | -3.777 | 8 | 34 | 8.436e-007 |
| 39 | 1 | -8.106 | 8 | 34 | 1.193e-007 |
| 40 | 1 | -3.592 | 10 | 36 | 8.572e-008 |

MultiStart completed the runs from all start points.

All 40 local solver runs converged with a positive local solver exit flag.

**xm =**

**-0.0093 1.5814**

**fm =**

**-8.1062**

**flagm =**

**1**

**outptm =**

**funcCount: 2171**

**localSolverTotal: 40**

**localSolverSuccess: 40**

**localSolverIncomplete: 0**

**localSolverNoSolution: 0**

**message: [1x128 char]**

**manyminms =**

**1x3 GlobalOptimSolution**

**Properties:**

**X**

**Fval**

**Exitflag**

**Output**

**X0**

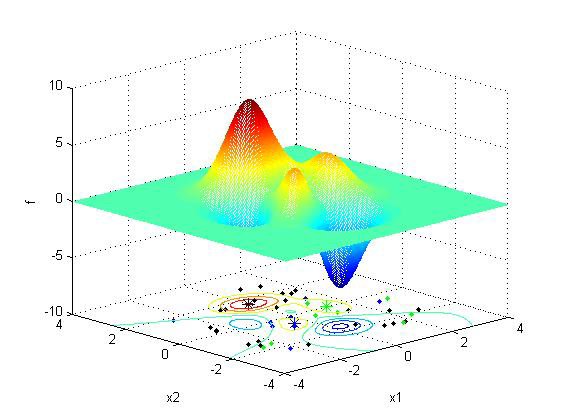


Рис. 16. Поиск максимумов функции myfun методом MultiStart

Как следует из текстового отчета и рис. 16, найдено три максимума, наибольший из которых равен 8,1062 и находится в точке (–0,0093 1,5814). Более точное представление о расположении стартовых точек и минимумов может дать контурная карта. Чтобы ее получить, изменим программу runglobal1\_3, уменьшив объем текстового отчета и убрав 3D-представление функции. Измененную программу runglobal1\_4 запускаем с той же начальной точки и получаем краткий отчет и контурную карту со всеми стартовыми точками и точками минимумов (рис. 17):

**>> x0=[3 -0.5]; runglobal1\_4(x0)**

MultiStart completed the runs from all start points.

All 40 local solver runs converged with a positive local solver exit flag.

**x =**

**-0.0093 1.5814**

**fval =**

**-8.1062**

**flag =**

**1**

**outpt =**

**funcCount: 2162**

**localSolverTotal: 40**

**localSolverSuccess: 40**

**localSolverIncomplete: 0**

**localSolverNoSolution: 0**

**message: [1x128 char]**

**manyminms =**

**1x3 GlobalOptimSolution**

**Properties:**

**X**

**Fval**

**Exitflag**

**Output**

**X0**

Чтобы получить значения всех найденных максимумов и их координат, обратимся к структуре manyminms:

>> [manyminms.Fval]

ans =

-8.1062 -3.7766 -3.5925

/значения функции с обратным знаком в трех

/максимумах

>> [manyminms.X]

ans =

Columns 1 through 5

-0.0093 1.5814 -0.4600 -0.6292 1.2857 /координаты трех

Column 6

-0.0048 /максимумов

Таким образом, задача решена успешно.

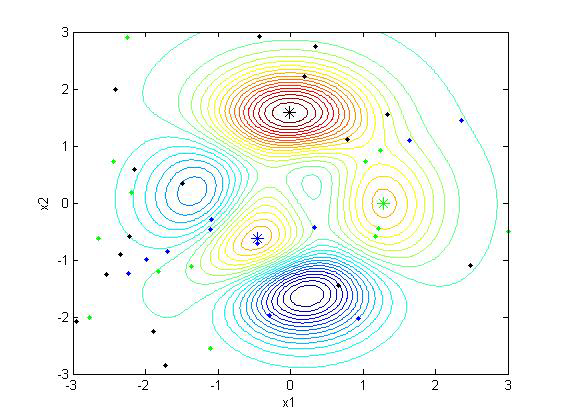


Рис. 17. Стартовые точки и максимумы (\*) на контурной карте

функции myfun

Пример 11.

Решим задачу из примера 10 при дополнительном ограничении

****

В программу runglobal1\_4 внесем изменения: при создании проблемы (problem) помимо границ зададим линейное ограничение, добавим функцию рисования этого ограничения, в команде run определим 50 стартовых точек и параметром StartPointsToRun в объекте ms потребуем использовать только те точки, которые удовлетворяют всем условиям. В итоге имеем следующую программу:

**function runglobal1\_5(x0)**

**[x,y]=meshgrid(-3:0.03:3);Z=myfun1(x,y);**

**c=contour(x,y,Z,30);**

**xlabel('x1');ylabel('x2');hold on; plot([-3 2], ...**

**[-1 3], 'g-','linewidth',2.3);**

**ms = MultiStart('StartPointsToRun','bounds-ineqs');**

**opts = optimset('Algorithm','interior-point');**

**problem = createOptimProblem('fmincon','x0',x0,...**

**'objective',@myfun2,'Aineq',[-4 5],'bineq',7,'lb', ...**

**[-3, -3],'ub',[3,3],'options',opts);**

**[x,fval,flag,outpt,manyminms] = run(ms,problem,50)**

**% Plot points**

**possColors = 'kbgcrm';**

**hold on**

**for i = 1:size(manyminms,2)**

**% Color of this line**

**cIdx = rem(i-1, length(possColors)) + 1;**

**color = possColors(cIdx);**

**% Plot start points**

**u = manyminms(i).X0;**

**x0ThisMin = reshape([u{:}], 2, length(u));**

**plot(x0ThisMin(1, :), x0ThisMin(2, :),'.', ...**

**'Color',color,'MarkerSize',12);**

**plot(manyminms(i).X(1), manyminms(i).X(2),'\*', ...**

**'Color', color, 'MarkerSize',10);**

**end % basin center marked with a \*, start points**

**% with dots**

**end**

Выполним эту программу:

>> x0=[3 -0.5]; runglobal1\_5(x0)

Выполнение завершается выдачей результатов решения задачи в виде рис. 18 и текстового отчета:

MultiStart completed the runs from all start points.

All 37 local solver runs converged with a positive

local solver exit flag.

x =

0.1265 1.5012

fval =

-7.8502

flag =

1

outpt =

funcCount: 1880

localSolverTotal: 37

localSolverSuccess: 37

localSolverIncomplete: 0

localSolverNoSolution: 0

message: [1x128 char]

manyminms =

1x3 GlobalOptimSolution

Properties:

X

Fval

Exitflag

Output

X0



Рис. 18. Расположение стартовых точек и максимумов

при линейном ограничении

Из рис. 18 видно, что все использованные стартовые точки лежат в допустимой области – это 37 точек из 50 заданных. Естественно, что глобальный максимум, равный 7,8502, оказался меньше, чем при отсутствии линейного неравенства, и потому лежит на границе этого неравенства. Как и в предыдущем примере, всего найдено три максимума, два из которых те же, что в варианте задачи без неравенства. Значения максимумов и их координаты получим из глобальной структуры manyminms:

>>manyminms.Fval

ans =

-7.8502

ans =

-3.7766

ans =

-3.5925

>>XX=[manyminms.X]

XX =

0.1265 1.5012 -0.4600 -0.6292 1.2857 -0.0048

В массиве ХХ представлены три пары точек, соответствующих координатам трех максимумов, а значения максимумов с обратным знаком – в переменных ans.