用最优化方法解决最优化问题的技术称为最优化技术，它包含两个方面的内容：

1） 建立数学模型 即用数学语言来描述最优化问题。模型中的数学关系式反映了最优化问题所要达到的目标和各种约束条件。

2） 数学求解 数学模型建好以后，选择合理的最优化方法进行求解。

最优化方法的发展很快，现在已经包含有多个分支，如线性规划、整数规划、非线性规划、动态规划、多目标规划等。

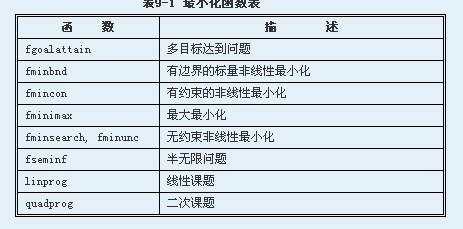
**9.1 概 述**

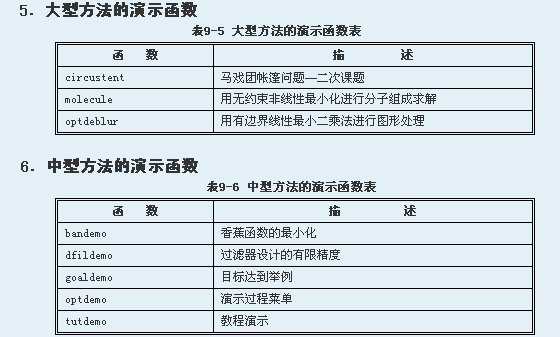
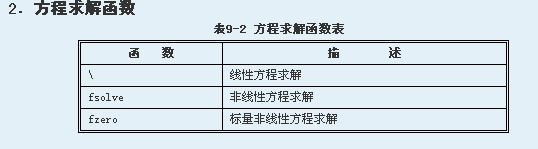
利用Matlab的优化工具箱，可以求解线性规划、非线性规划和多目标规划问题。具体而言，包括线性、非线性最小化，最大最小化，二次规划，半无限问题，线性、非线性方程（组）的求解，线性、非线性的最小二乘问题。另外，该工具箱还提供了线性、非线性最小化，方程求解，曲线拟合，二次规划等问题中大型课题的求解方法，为优化方法在工程中的实际应用提供了更方便快捷的途径。

**9.1.1 优化工具箱中的函数**

优化工具箱中的函数包括下面几类：

**1．最小化函数**





**9.1.3 参数设置**

利用optimset函数，可以创建和编辑参数结构；利用optimget函数，可以获得options优化参数。

● optimget函数

**功能**：获得options优化参数。

**语法**：

val = optimget(options,'param')

val = optimget(options,'param',default)

**描述**：

val = optimget(options,'param') 返回优化参数options中指定的参数的值。只需要用参数开头的字母来定义参数就行了。

val = optimget(options,'param',default) 若options结构参数中没有定义指定参数，则返回缺省值。注意，这种形式的函数主要用于其它优化函数。

**举例**：

1．       下面的命令行将显示优化参数options返回到my\_options结构中：

val = optimget(my\_options,'Display')

2．       下面的命令行返回显示优化参数options到my\_options结构中（就象前面的例子一样），但如果显示参数没有定义，则返回值'final':

optnew = optimget(my\_options,'Display','final');

**参见**：

optimset

● optimset函数

**功能**：创建或编辑优化选项参数结构。

**语法**：

options = optimset('param1',value1,'param2',value2,...)

optimset

options = optimset

options = optimset(optimfun)

options = optimset(oldopts,'param1',value1,...)

options = optimset(oldopts,newopts)

**描述**：

options = optimset('param1',value1,'param2',value2,...) 创建一个称为options的优化选项参数，其中指定的参数具有指定值。所有未指定的参数都设置为空矩阵[]（将参数设置为[]表示当options传递给优化函数时给参数赋缺省值）。赋值时只要输入参数前面的字母就行了。

optimset函数没有输入输出变量时，将显示一张完整的带有有效值的参数列表。

options = optimset (with no input arguments) 创建一个选项结构options，其中所有的元素被设置为[]。

options = optimset(optimfun) 创建一个含有所有参数名和与优化函数optimfun相关的缺省值的选项结构options。

options = optimset(oldopts,'param1',value1,...) 创建一个oldopts的拷贝，用指定的数值修改参数。

options = optimset(oldopts,newopts) 将已经存在的选项结构oldopts与新的选项结构newopts进行合并。newopts参数中的所有元素将覆盖oldopts参数中的所有对应元素。

**举例**：

1．下面的语句创建一个称为options的优化选项结构，其中显示参数设为'iter'，TolFun参数设置为1e-8:

options = optimset('Display','iter','TolFun',1e-8)

2．下面的语句创建一个称为options的优化结构的拷贝，改变TolX参数的值，将新值保存到optnew参数中:

optnew = optimset(options,'TolX',1e-4);

3．下面的语句返回options优化结构，其中包含所有的参数名和与fminbnd函数相关的缺省值：

options = optimset('fminbnd')

4．若只希望看到fminbnd函数的缺省值，只需要简单地键入下面的语句就行了：

optimset fminbnd

或者输入下面的命令，其效果与上面的相同：

optimset('fminbnd')

**参见**：

optimget

**9.1.4 模型输入时需要注意的问题**

使用优化工具箱时，由于优化函数要求目标函数和约束条件满足一定的格式，所以需要用户在进行模型输入时注意以下几个问题：

1.目标函数最小化

优化函数fminbnd、fminsearch、fminunc、fmincon、fgoalattain、fminmax和lsqnonlin都要求目标函数最小化，如果优化问题要求目标函数最大化，可以通过使该目标函数的负值最小化即-f(x)最小化来实现。近似地，对于quadprog函数提供-H和-f，对于linprog函数提供-f。

2.约束非正

优化工具箱要求非线性不等式约束的形式为Ci(x)≤0，通过对不等式取负可以达到使大于零的约束形式变为小于零的不等式约束形式的目的，如Ci(x)≥0形式的约束等价于- Ci(x)≤0；Ci(x)≥b形式的约束等价于- Ci(x)+b≤0。

3.避免使用全局变量

**9.1.5 @（函数句柄）函数**

MATLAB6.0中可以用@函数进行函数调用。@函数返回指定MATLAB函数的句柄，其调用格式为：

handle = @function

利用@函数进行函数调用有下面几点好处：

●    用句柄将一个函数传递给另一个函数；

●    减少定义函数的文件个数；

●    改进重复操作；

●    保证函数计算的可靠性。

下面的例子为humps函数创建一个函数句柄，并将它指定为fhandle变量。

fhandle = @humps;

同样传递句柄给另一个函数，也将传递所有变量。本例将刚刚创建的函数句柄传递给fminbnd函数，然后在区间[0.3,1]上进行最小化。

x = fminbnd (@humps, 0.3, 1)

x =

0.6370

**9.2 最小化问题**

**9.2.1 单变量最小化**

**9.2.1.1 基本数学原理**

本节讨论只有一个变量时的最小化问题，即一维搜索问题。该问题在某些情况下可以直接用于求解实际问题，但大多数情况下它是作为多变量最优化方法的基础在应用，因为进行多变量最优化要用到一维搜索法。该问题的数学模型为：

其中，**x**,**x1**,和**x2**为标量，**f**(**x**)为函数，返回标量。

该问题的搜索过程可用下式表达：

其中xk为本次迭代的值，d为搜索方向，α为搜索方向上的步长参数。所以一维搜索就是要利用本次迭代的信息来构造下次迭代的条件。

求解单变量最优化问题的方法有很多种，根据目标函数是否需要求导，可以分为两类，即直接法和间接法。直接法不需要对目标函数进行求导，而间接法则需要用到目标函数的导数。

**1．直接法**

常用的一维直接法主要有消去法和近似法两种。

（1）消去法 该法利用单峰函数具有的消去性质进行反复迭代，逐渐消去不包含极小点的区间，缩小搜索区间，直到搜索区间缩小到给定的允许精度为止。一种典型的消去法为黄金分割法(Golden Section Search)。黄金分割法的基本思想是在单峰区间内适当插入两点，将区间分为三段，然后通过比较这两点函数值的大小来确定是删去最左段还是最右段，或同时删去左右两段保留中间段。重复该过程使区间无限缩小。插入点的位置放在区间的黄金分割点及其对称点上，所以该法称为黄金分割法。该法的优点是算法简单，效率较高，稳定性好。

（2）多项式近似法 该法用于目标函数比较复杂的情况。此时寻找一个与它近似的函数代替目标函数，并用近似函数的极小点作为原函数极小点的近似。常用的近似函数为二次和三次多项式。

二次内插涉及到形如下式的二次函数数据拟合问题：

其中步长极值为：

然后只要利用三个梯度或函数方程组就可以确定系数a和b，从而可以确定α\*。得到该值以后，进行搜索区间的收缩。在缩短的新区间中，重新安排三点求出下一次的近似极小点α\*，如此迭代下去，直到满足终止准则为止。其迭代公式为：

其中

二次插值法的计算速度比黄金分割法的快，但是对于一些强烈扭曲或可能多峰的函数，该法的收敛速度会变得很慢，甚至失败。

**2．间接法**

间接法需要计算目标函数的导数，优点是计算速度很快。常见的间接法包括牛顿切线法、对分法、割线法和三次插值多项式近似法等。优化工具箱中用得较多的是三次插值法。

三次插值的基本思想与二次插值的一致，它是用四个已知点构造一个三次多项式P3(x)，用它逼近函数f(x)，以P3(x)的极小点作为f(x)的近似极小点。一般讲，三次插值法比二次插值法的收敛速度要快些，但每次迭代需要计算两个导数值。

三次插值法的迭代公式为

其中

如果函数的导数容易求得，一般来说首先考虑使用三次插值法，因为它具有较高的效率。对于只需要计算函数值的方法中，二次插值法是一个很好的方法，它的收敛速度较快，尤其在极小点所在区间较小时尤其如此。黄金分割法则是一种十分稳定的方法，并且计算简单。由于以上原因，Matlab优化工具箱中使用得较多的方法是二次插值法、三次插值法、二次、三次混合插值法和黄金分割法。

**9.2.1.2 相关函数介绍**

fminbnd

**功能**：找到固定区间内单变量函数的最小值。

**语法**：

x = fminbnd(fun,x1,x2)

x = fminbnd(fun,x1,x2,options)

x = fminbnd(fun,x1,x2,options,P1,P2,...)

[x,fval] = fminbnd(...)

[x,fval,exitflag] = fminbnd(...)

[x,fval,exitflag,output] = fminbnd(...)

**描述**：

fminbnd求取固定区间内单变量函数的最小值。

x = fminbnd(fun,x1,x2)返回区间{x1，x2}上fun参数描述的标量函数的最小值x。

x = fminbnd(fun,x1,x2,options)用options参数指定的优化参数进行最小化。

x = fminbnd(fun,x1,x2,options,P1,P2,...)提供另外的参数P1,P2等,传输给目标函数fun。如果没有设置options选项，则令options=[]。

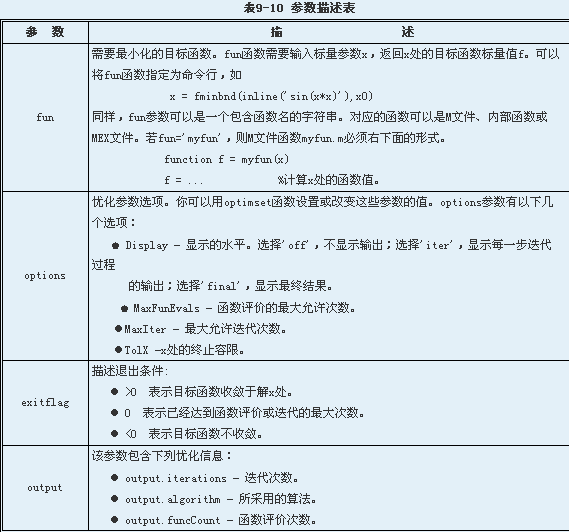
[x,fval] = fminbnd(...)返回解x处目标函数的值。

[x,fval,exitflag] = fminbnd(...)返回exitflag值描述fminbnd函数的退出条件。

[x,fval,exitflag,output] = fminbnd(...)返回包含优化信息的结构输出。

**变量**：

函数的输入变量在表9-7中进行描述，输出变量在表9-8中描述。与fminbnd函数相关的细节内容包含在fun,options,exitflag和output等参数中，如表9-10所示。

**算法**：

fminbnd是一个M文件。其算法基于黄金分割法和二次插值法。文献[1]中给出了实现同样算法的Fortran程序。

**局限性**：

1．目标函数必须是连续的。

2．fminbnd函数可能只给出局部最优解。

3．当问题的解位于区间边界上时，fminbnd函数的收敛速度常常很慢。此时，fmincon函数的计算速度更快，计算精度更高。

4．fminbnd函数只用于实数变量。

**参见**：

fminsearch, fmincon, fminunc, optimset, inline

**文献**：

[1]   Forsythe, G.E., M.A. Malcolm, and C.B. Moler, Computer Methods for Mathematical Computations, Prentice Hall, 1976.

**9.2.1.3 应用实例**

**[例一]** 在区间（0，2π）上求函数sin(x)的最小值：

x = fminbnd(@sin,0,2\*pi)

x =

4.7124

所以区间（0，2π）上函数sin(x)的最小值点位于x=4.7124处。

最小值处的函数值为：

y = sin(x)

y =

-1.0000

磁盘中该问题的M文件名为opt21\_1.m。

**[例三]** 对边长为3m的正方形铁板，在四个角处剪去相等的正方形以制成方形无盖水槽，问如何剪法使水槽的容积最大？

假设剪去的正方形的边长为x，则水槽的容积为

现在要求在区间（0，1.5）上确定一个**x**，使 最大化。因为优化工具箱中要求目标函数最小化，所以需要对目标函数进行转换，即要求 最小化。

首先编写M文件opt21\_3o.m:

function f = myfun(x)

f = -(3-2\*x).^2 \* x;

然后调用fminbnd函数(磁盘中M文件名为opt21\_3.m)：

x = fminbnd(@opt21\_3o,0,1.5)

得到问题的解：

x =

0.5000

即剪掉的正方形的边长为0.5m时水槽的容积最大。

水槽的最大容积计算：

y = optim2(x)

y =

-2.0000

所以水槽的最大容积为2.0000m3。

**9.2.2 线性规划**

**9.2.2.1 基本数学原理**

线性规划是处理线性目标函数和线性约束的一种较为成熟的方法，目前已经广泛应用于军事、经济、工业、农业、教育、商业和社会科学等许多方面。线性规划问题的标准形式是：

或

写成矩阵形式为：

其中，**0**为n维列向量。

线性规划的标准形式要求目标函数最小化，约束条件取等式，变量非负。不符合这几个条件的线性模型要首先转化成标准形。

线性规划的求解方法主要是单纯形法（Simple Method），该法由Dantzig于1947年提出，以后经过多次改进。单纯形法是一种迭代算法，它从所有基本可行解的一个较小部分中通过迭代过程选出最优解。其迭代过程的一般描述为：

1． 将线性规划化为典范形式，从而可以得到一个初始基本可行解x(0)(初始顶点)，将它作为迭代过程的出发点，其目标值为z(x(0))。

2． 寻找一个基本可行解x(1)，使z(x(1))≤z(x(0))。方法是通过消去法将产生x(0)的典范形式化为产生x(1)的典范形式。

3． 继续寻找较好的基本可行解x(2)，x(3)，…，使目标函数值不断改进，即z(x(1))≥z(x(2)) ≥z(x(3)) ≥…。当某个基本可行解再也不能被其它基本可行解改进时，它就是所求的最优解。

Matlab优化工具箱中采用的是投影法，它是单纯形法的一种变种。

**9.2.2.2 相关函数介绍**

linprog函数

**功能**：求解线性规划问题。

**数学模型**：

其中**f**, **x**, **b**, **beq**, **lb**和**ub**为向量，**A** 和**Aeq**为矩阵。

**语法**：

x = linprog(f,A,b,Aeq,beq)

x = linprog(f,A,b,Aeq,beq,lb,ub)

x = linprog(f,A,b,Aeq,beq,lb,ub,x0)

x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)

[x,fval] = linprog(...)

[x,fval,exitflag] = linprog(...)

[x,fval,exitflag,output] = linprog(...)

[x,fval,exitflag,output,lambda] = linprog(...)

**描述**：

x = linprog(f,A,b)求解问题 min f'\*x，约束条件为A\*x <= b。

x = linprog(f,A,b,Aeq,beq)求解上面的问题，但增加等式约束，即Aeq\*x = beq。若没有不等式存在，则令A=[]、b=[]。

x = linprog(f,A,b,Aeq,beq,lb,ub)定义设计变量x的下界lb和上界ub，使得x始终在该范围内。若没有等式约束，令Aeq=[]、beq=[]。

x = linprog(f,A,b,Aeq,beq,lb,ub,x0)设置初值为x0。该选项只适用于中型问题，缺省时大型算法将忽略初值。

x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)用options指定的优化参数进行最小化。

[x,fval] = linprog(...) 返回解x处的目标函数值fval。

[x,lambda,exitflag] = linprog(...)返回exitflag值，描述函数计算的退出条件。

[x,lambda,exitflag,output] = linprog(...) 返回包含优化信息的输出变量output。

[x,fval,exitflag,output,lambda] = linprog(...) 将解x处的拉格朗日乘子返回到lambda参数中。

**变量**：

lambda参数

lambda参数是解x处的拉格朗日乘子。它有以下一些属性：

       lambda.lower –lambda的下界。

       lambda.upper –lambda的上界。

       lambda.ineqlin –lambda的线性不等式。

       lambda.eqlin –lambda的线性等式。

其它参数意义同前。

**算法**：

大型优化算法 大型优化算法采用的是LIPSOL法，该法在进行迭代计算之前首先要进行一系列的预处理。

中型优化算法 linprog函数使用的是投影法，就象quadprog函数的算法一样。linprog函数使用的是一种活动集方法，是线性规划中单纯形法的变种。它通过求解另一个线性规划问题来找到初始可行解。

**诊断**：

大型优化问题 算法的第一步涉及到一些约束条件的预处理问题。有些问题可能导致linprog函数退出，并显示不可行的信息。在本例中，exitflag参数将被设为负值以表示优化失败。

若Aeq参数中某行的所有元素都为零，但Beq参数中对应的元素不为零，则显示以下退出信息：

Exiting due to infeasibility: an all zero row in the constraint matrix does not have a zero in corresponding right hand size entry.

若x的某一个元素没在界内，则给出以下退出信息：

Exiting due to infeasibility: objective f'\*x is unbounded below.

若Aeq参数的某一行中只有一个非零值，则x中的相关值称为奇异变量。这里，x中该成分的值可以用Aeq和Beq算得。若算得的值与另一个约束条件相矛盾，则给出以下退出信息：

Exiting due to infeasibility: Singleton variables in equality constraints are not feasible.

若奇异变量可以求解但其解超出上界或下界，则给出以下退出信息：

Exiting due to infeasibility: singleton variables in the equality constraints are not within bounds.

**9.2.2.3 应用实例**

**[ [例二]** 生产决策问题

某厂生产甲乙两种产品，已知制成一吨产品甲需用资源A 3吨，资源B 4m3；制成一吨产品乙需用资源A 2吨，资源B 6m3，资源C 7个单位。若一吨产品甲和乙的经济价值分别为7万元和5万元，三种资源的限制量分别为90吨、200m3和210个单位，试决定应生产这两种产品各多少吨才能使创造的总经济价值最高？

令生产产品甲的数量为x1，生产产品乙的数量为x2。由题意可以建立下面的模型：

该模型中要求目标函数最大化，需要按照Matlab的要求进行转换，即目标函数为

首先输入下列系数：

f = [-7;-5];

A = [3 2

4 6

0 7];

b = [90; 200; 210];

lb = zeros(2,1);

然后调用linprog函数：

[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb)

x =

14.0000

24.0000

fval =

-218.0000

exitflag =

1

output =

iterations: 5

cgiterations: 0

algorithm: 'lipsol'

lambda =

ineqlin: [3x1 double]

eqlin: [0x1 double]

upper: [2x1 double]

lower: [2x1 double]

由上可知，生产甲种产品14吨、乙种产品24吨可使创建的总经济价值最高。最高经济价值为218万元。exitflag=1表示过程正常收敛于解x处。

磁盘中本问题的M文件为opt22\_2.m。

**[例三]** 投资问题

某单位有一批资金用于四个工程项目的投资，用于各工程项目时所得到得净收益（投入资金的百分比）如下表所示：

**表9-11 工程项目收益表**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **工 程 项 目** | **A** | **B** | **C** | **D** |
| 收益（%） | 15 | 10 | 8 | 12 |

由于某种原因，决定用于项目A的投资不大于其它各项投资之和；而用于项目B和C的投资要大于项目D的投资。试确定使该单位收益最大的投资分配方案。

用x1、x2、x3和x4分别代表用于项目A、B、C和D的投资百分数，由于各项目的投资百分数之和必须等于100%，所以

x1+x2+x3+x4=1

据题意，可以建立下面的数学模型：

将它转换为标准形式：

然后进行求解：

首先输入下列系数：

f = [-0.15;-0.1;-0.08;-0.12];

A = [1 -1 -1 -1

0 -1 -1 1];

b = [0; 0];

Aeq=[1 1 1 1];

beq=[1];

lb = zeros(4,1);

然后调用linprog函数：

[x,fval,exitflag,output,lambda] = linprog(f,A,b,Aeq,beq,lb)；

x =

0.5000

0.2500

0.0000

0.2500

fval =

-0.1300

exitflag =

1

可见，四个项目的投资百分数分别为0.50、0.25、0.00和0.25时可使该单位获得最大的收益。最大收益为13%。过程正常收敛。

磁盘中本问题的M文件为opt22\_3.m。

**[例四]** 工件加工任务分配问题

某车间有两台机床甲和乙，可用于加工三种工件。假定这两台机床的可用台时数分别为700和800，三种工件的数量分别为300、500和400，且已知用三种不同机床加工单位数量的不同工件所需的台时数和加工费用（如表 所示），问怎样分配机床的加工任务，才能既满足加工工件的要求，又使总加工费用最低？

设在甲机床上加工工件1、2和3的数量分别为x1、x2和x3，在乙机床上加工工件1、2和3的数量分别为x4、x5和x6。根据三种工种的数量限制，有

x1+x4=300 (对工件1)

x2+x5=500 (对工件2)

x3+x6=400 (对工件3)

再根据机床甲和乙的可用总台时限制，可以得到其它约束条件。以总加工费用最少为目标函数，组合约束条件，可以得到下面的数学模型：

首先输入下列系数：

f = [13;9;10;11;12;8];

A = [0.4 1.1 1 0 0 0

0 0 0 0.5 1.2 1.3];

b = [700; 800];

Aeq=[1 0 0 1 0 0

0 1 0 0 1 0

0 0 1 0 0 1];

beq=[300 500 400];

lb = zeros(6,1);

然后调用linprog函数：

[x,fval,exitflag,output,lambda] = linprog(f,A,b,Aeq,beq,lb);

x =

0.0000

500.0000

0.0000

300.0000

0.0000

400.0000

fval =

1.1000e+004

exitflag =

1

可见，在甲机床上加工500个工件2，在乙机床上加工300个工件1、加工400个工件3可在满足条件的情况下使总加工费最小。最小费用为11000元。收敛正常。

磁盘中本问题的M文件为opt22\_4.m。

**[例五]** 裁料问题

在某建筑工程施工中需要制作10000套钢筋，每套钢筋由2.9m、2.1m和1.5m三种不同长度的钢筋各一根组成，它们的直径和材质不同。目前在市场上采购到的同类钢筋的长度每根均为7.4m，问应购进多少根7.4m长的钢筋才能满足工程的需要？

首先分析共有多少种不同的套裁方法，该问题的可能材料方案如表9-13所示。

 设以**xi**(i=1,2,…,8)表示按第i种裁料方案下料的原材料数量，则可得该问题的数学模型为：

首先输入下列系数：

f = [1;1;1;1;1;1;1;1];

Aeq=[2 0 0 0 0 0 0 0

0 2 1 0 3 2 1 0

1 0 1 3 0 2 3 4];

beq=[10000 10000 10000];

lb = zeros(8,1);

然后调用linprog函数：

[x,fval,exitflag,output,lambda] = linprog(f,[],[],Aeq,beq,lb);

x =

1.0e+003 \*

5.0000

0.0000

0.0000

0.0000

1.6667

2.5000

0.0000

0.0000

fval =

9.1667e+003

所以最节省的情况需要9167根7.4m长的钢筋，其中第一种方案使用5000根，第五种方案使用1667根，第六种方案使用2500根。

磁盘中本问题的M文件为opt22\_5.m。

**[例六]** 工作人员计划安排问题

某昼夜服务的公共交通系统每天各时间段（每4小时为一个时间段）所需的值班人数如表 所示，这些值班人员在某一时段开始上班后要连续工作8个小时（包括轮流用膳时间），问该公交系统至少需要多少名工作人员才能满足值班的需要？

 设xi为第i个时段开始上班的人员数，据题意建立下面的数学模型：

需要对前面六个约束条件进行形式变换，是不等式为非正不等式。只需要在不等式两侧取负即可。

首先输入下列系数：

f = [1;1;1;1;1;1];

A=[-1 0 0 0 0 -1

-1 -1 0 0 0 0

0 -1 -1 0 0 0

0 0 -1 -1 0 0

0 0 0 -1 -1 0

0 0 0 0 -1 -1];

b=[-60;-70;-60;-50;-20;-30];

lb = zeros(6,1);

然后调用linprog函数：

[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb);

x =

41.9176

28.0824

35.0494

14.9506

9.8606

20.1394

fval =

150.0000

exitflag =

1

可见，只要六个时段分别安排42人、28人、35人、15人、10人和20人就可以满足值班的需要。共计150人。计算收敛。

磁盘中本问题的M文件为opt22\_6.m。

**[例七]** 厂址选择问题

考虑A、B、C三地，每地都出产一定数量的原料，也消耗一定数量的产品（见表9-15）。已知制成每吨产品需3吨原料，各地之间的距离为：A-B：150km，A-C：100km，B-C：200km。假定每万吨原料运输1km的运价是5000元，每万吨产品运输1km的运价是6000元。由于地区条件的差异，在不同地点设厂的生产费用也不同。问究竟在哪些地方设厂，规模多大，才能使总费用最小？另外，由于其它条件限制，在B处建厂的规模（生产的产品数量）不能超过5万吨。

 令xij为由i地运到j地的原料数量（万吨），yij为由i地运往j地的产品数量（万吨），i,j=1,2,3（分别对应A、B、C三地）。根据题意，可以建立问题的数学模型（其中目标函数包括原材料运输费、产品运输费和生产费）：

首先输入下列系数：

f = [75;75;50;50;100;100;150;240;210;120;160;220];

A=[1 -1 1 -1 0 0 3 3 0 0 0 0

-1 1 0 0 1 -1 0 0 3 3 0 0

0 0 -1 1 -1 1 0 0 0 0 3 3

0 0 0 0 0 0 0 0 1 1 0 0];

b=[20;16;24;5];

Aeq=[0 0 0 0 0 0 1 0 1 0 1 0

0 0 0 0 0 0 0 1 0 1 0 1];

beq=[7;13];

lb = zeros(12,1);

然后调用linprog函数：

[x,fval,exitflag,output,lambda] = linprog(f,A,b,Aeq,beq,lb);

x =

0.0000

1.0000

0.0000

0.0000

0.0000

0.0000

7.0000

0.0000

0.0000

5.0000

0.0000

8.0000

fval =

3.4850e+003

exitflag =

1

要使总费用最小，需要B地向A地运送1万吨，A、B、C三地的建厂规模分别为7万吨、5万吨和8万吨。最小总费用为3485元。

磁盘中本问题的M文件为opt22\_7.m。

**[例八]** 确定职工编制问题

某厂每日八小时的产量不低于1800件。为了进行质量控制，计划聘请两种不同水平的检验员。一级检验员的标准为：速度 25件/小时，正确率 98%，计时工资 4元/小时；二级检验员的标准为：速度 15件/小时，正确率 95%，计时工资 3元/小时。检验员每错检一次，工厂要损失2元。现有可供厂方聘请的检验员人数为一级8人和二级10人。为使总检验费用最省，该工厂应聘一级、二级检验员各多少名？

设需要一级和二级检验员的人数分别为x1名和x2名，由题意可以建立下面的模型：

利用Matlab进行求解之前需要将第三个约束条件进行转换，两边取负以后得到

首先输入下列系数：

f = [40;36];

A=[1 0

0 1

-5 -3];

b=[8;10;-45];

lb = zeros(2,1);

然后调用linprog函数：

[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb);

x =

8.0000

1.6667

fval =

380.0000

exitflag =

1

可见，招聘一级检验员8名、二级检验员2名可使总检验费最省，约为380.00元。计算收敛。

磁盘中本问题的M文件为opt22\_8.m。

**[例九]** 生产计划的最优化问题

某工厂生产A和B两种产品，它们需要经过三种设备的加工，其工时如表9-16所示。设备一、二和三每天可使用的时间分别不超过12、10和8小时。产品A和B的利润随市场的需求有所波动，如果预测未来某个时期内A和B的利润分别为4和3千元/吨，问在那个时期内，每天应安排产品A、B各多少吨，才能使工厂获利最大？

 设每天应安排生产产品A和B分别为x1吨和x2吨，由题意建立下面的数学模型：

首先转换目标函数为标准形式：

输入下列系数：

f = [-4;-3];

A=[3 4

3 3

4 2];

b=[12;10;8];

lb = zeros(2,1);

然后调用linprog函数：

[x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb);

x =

0.8000

2.4000

fval =

-10.4000

所以，每天生产A产品0.80吨、B产品2.40吨可使工厂获得最大利润。

磁盘中本问题的M文件为opt22\_9.m。

**9.2.3 无约束非线性规划问题**

**9.2.3.1 基本数学原理**

无约束最优化问题在实际应用中也比较常见，如工程中常见的参数反演问题。另外，许多有约束最优化问题可以转化为无约束最优化问题进行求解。

求解无约束最优化问题的方法主要有两类，即直接搜索法（Search method）和梯度法（Gradient method）。

直接搜索法适用于目标函数高度非线性，没有导数或导数很难计算的情况，由于实际工程中很多问题都是非线性的，直接搜索法不失为一种有效的解决办法。常用的直接搜索法为单纯形法，此外还有Hooke-Jeeves搜索法、Pavell共轭方向法等，其缺点是收敛速度慢。

在函数的导数可求的情况下，梯度法是一种更优的方法，该法利用函数的梯度（一阶导数）和Hessian矩阵（二阶导数）构造算法，可以获得更快的收敛速度。函数f(x)的负梯度方向-▽f(x)即反映了函数的最大下降方向。当搜索方向取为负梯度方向时称为最速下降法。当需要最小化的函数有一狭长的谷形值域时，该法的效率很低，如Rosenbrock函数

f(x)=100(x1-x22)2+(1-x1)2

它的最小值解为x=[1,1]，最小值为f(x)=0。下图是该函数的等值线图，图中还显示了从初值[-1.9，2]出发向最小值前进的路径。迭代1000次以后终止，此时距最小值仍有相当长的距离。图中的黑色区域是该法在谷的两侧不断进行“之”字形搜索形成的。

**图9-1 香蕉函数的等值线图及最速下降法的搜索路径**

这种类型的函数又称为香蕉函数。

常见的梯度法有最速下降法、Newton法、Marquart法、共轭梯度法和拟牛顿法（Quasi-Newton method）等。

在所有这些方法中，用得最多的是拟牛顿法，这些方法在每次迭代过程中建立曲率信息，构成下式得二次模型问题：

其中，Hessian矩阵H为一正定对称矩阵，c为常数向量，b为常数。对x求偏导数可以获得问题的最优解

解x\*可写作：

拟牛顿法包括两个阶段，即

●    确定搜索方向

●    一维搜索阶段

1．Hessian矩阵的更新

牛顿法由于需要多次计算Hessian矩阵，计算量很大，而拟牛顿法则通过构建一个Hessian矩阵的近似矩阵来避开这个问题。

在优化工具箱中，通过将options参数HessUpdate设置为BFGS或DFP来决定搜索方向。当Hessian矩阵H始终保持正定时，搜索方向就总是保持为下降方向。

Hessian矩阵的方法很多，对于求解一般问题，Broyden,Hetcher,Goldfarb和Shanno的方法（简称BFGS法）是最有效的。BFGS法的计算公式为：

其中

作为初值，H0可以设为任意对称正定矩阵。

另一个有名的构造近似Hessian矩阵的方法是DFP（Daridon-Fletcher-Powell）法。该法的计算公式与BFGS法的形式一样，只是qk替换为sk。

梯度信息可以是用解析的方法得到，也可以是用有限差分的方法通过求偏导数得到。

在每一个主要的迭代过程中，在下式所示的方向上进行一维搜索：

图9-2演示了用拟牛顿法时Rosenbrock函数的求解路径。可见，利用该法，只需要140次迭代就可以达到最小值解，比前面利用最速下降法要快得多。

**图9-2 拟牛顿法的搜索路径**

2．一维搜索

工具箱中有两套方案进行一维搜索。当梯度值可以直接得到时，用三次插值的方法进行一维搜索，当梯度值不能直接得到时，采用二次、三次混合插值法。

**9.2.3.2 相关函数介绍**

fminunc函数

**功能**：求多变量无约束函数的最小值。

**数学模型**：

其中，**x**为一向量，**f(x)**为一函数，返回标量。

**语法格式**：

x = fminunc(fun,x0)

x = fminunc(fun,x0,options)

x = fminunc(fun,x0,options,P1,P2,...)

[x,fval] = fminunc(...)

[x,fval,exitflag] = fminunc(...)

[x,fval,exitflag,output] = fminunc(...)

[x,fval,exitflag,output,grad] = fminunc(...)

[x,fval,exitflag,output,grad,hessian] = fminunc(...)

**描述**：

fminunc给定初值，求多变量标量函数的最小值。常用于无约束非线性最优化问题。

x = fminunc(fun,x0)给定初值x0，求fun函数的局部极小点x。x0可以是标量、向量或矩阵。

x = fminunc(fun,x0,options)用options参数中指定的优化参数进行最小化。

x = fminunc(fun,x0,options,P1,P2,...)将问题参数p1、p2等直接输给目标函数fun，将options参数设置为空矩阵，作为options参数的缺省值。

[x,fval] = fminunc(...)将解x处目标函数的值返回到fval参数中。

[x,fval,exitflag] = fminunc(...)返回exitflag值，描述函数的输出条件。

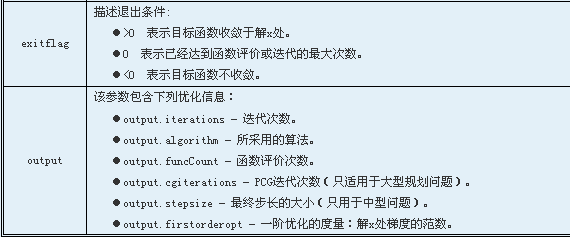
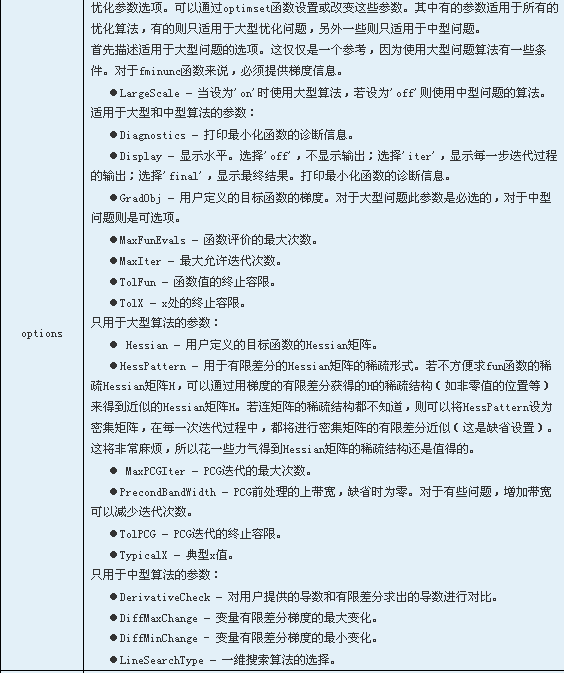
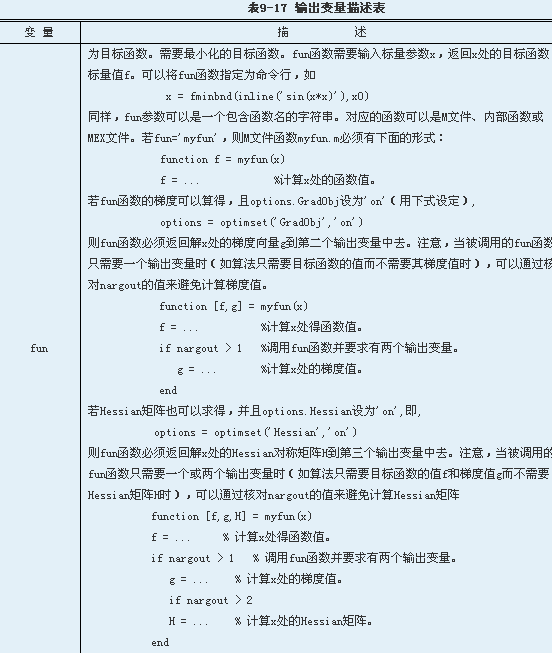
[x,fval,exitflag,output] = fminunc(...)返回包含优化信息的结构输出。

[x,fval,exitflag,output,grad] = fminunc(...)将解x处fun函数的梯度值返回到grad参数中。

[x,fval,exitflag,output,grad,hessian] = fminunc(...)将解x处目标函数的Hessian矩阵信息返回到hessian参数中。

**变量**：

表9-7中为输入变量，表9-8中为输出变量的描述。

**注意**：

1．对于求解平方和的问题，fminunc函数不是最好的选择，用lsqnonlin函数效果更佳。

2．使用大型方法时，必须通过将options.GradObj设置为'on'来提供梯度信息，否则将给出警告信息。

**算法**：

大型优化算法 若用户在fun函数中提供梯度信息，则缺省时函数将选择大型优化算法，该算法是基于内部映射牛顿法的子空间置信域法，理论描述可参见文献[8],[9]。计算中的每一次迭代涉及到用PCG法求解大型线性系统得到的近似解。

中型优化算法 此时fminunc函数的参数options.LargeScale设置为'off'。该算法采用的是基于二次和三次混合插值一维搜索法的BFGS拟牛顿法。该法通过BFGS公式来更新Hessian矩阵。通过将HessUpdate参数设置为'dfp'，可以用DFP公式来求得Hessian矩阵逆的近似。通过将HessUpdate参数设置为'steepdesc'，可以用最速下降法来更新Hessian矩阵。但一般不建议使用最速下降法。

缺省时的一维搜索算法，当options.LineSearchType设置为'quadcubic'时,将采用二次和三次混合插值法。将options.LineSearchType设置为'cubicpoly'时，将采用三次插值法。第二种方法需要的目标函数计算次数更少，但梯度的计算次数更多。这样，如果提供了梯度信息，或者能较容易地算得，则三次插值法是更佳的选择。

**局限性**：

1．          目标函数必须是连续的。fminunc函数有时会给出局部最优解。

2．          fminunc函数只对实数进行优化，即x必须为实数，而且f(x)必须返回实数。当x为复数时，必须将它分解为实部和虚部。

3．          在使用大型算法时，用户必须在fun函数中提供梯度（options参数中GradObj属性必须设置为'on'）。

4．          目前，若在fun函数中提供了解析梯度，则options参数DerivativeCheck不能用于大型算法以比较解析梯度和有限差分梯度。通过将options参数的MaxIter 属性设置为0来用中型方法核对导数。然后重新用大型方法求解问题。

**参见**：

fminsearch, optimset, inline

fminsearch函数

**功能**：求解多变量无约束函数的最小值。

**数学模型**：

其中x为向量，f(x)为函数，返回标量。

**语法**：

x = fminsearch(fun,x0)

x = fminsearch(fun,x0,options)

x = fminsearch(fun,x0,options,P1,P2,...)

[x,fval] = fminsearch(...)

[x,fval,exitflag] = fminsearch(...)

[x,fval,exitflag,output] = fminsearch(...)

**描述**：

fminsearch 求解多变量无约束函数的最小值。该函数常用于无约束非线性最优化问题。

x = fminsearch(fun,x0) 初值为x0，求fun函数的局部极小点x。x0可以是标量、向量或矩阵。

x = fminsearch(fun,x0,options)用options参数指定的优化参数进行最小化。

x = fminsearch(fun,x0,options,P1,P2,...) 将问题参数p1、p2等直接输给目标函数fun，将options参数设置为空矩阵，作为options参数的缺省值。

[x,fval] = fminsearch(...)将x处的目标函数值返回到fval参数中。

[x,fval,exitflag] = fminsearch(...)返回exitflag值，描述函数的退出条件。

[x,fval,exitflag,output] = fminsearch(...)返回包含优化信息的输出参数output。

**变量**：

各变量的意义同前。

**算法**：

fminsearch使用单纯形法进行计算。

对于求解二次以上的问题，fminsearch函数比fminunc函数有效。但是，当问题为高度非线性时，fminsearch函数更具稳健性。

**局限性**：

1．          应用fminsearch函数可能会得到局部最优解。

2．          fminsearch函数只对实数进行最小化，即x必须由实数组成，f(x)函数必须返回实数。如果x时复数，必须将它分为实数部和虚数部两部分。

**注意**:

fminsearch函数不适合求解平方和问题，用lsqnonlin函数更好一些。

**参见**：

fminbnd, fminunc, optimset, inline

**9.2.4 二次规划问题**

**9.2.4.1 基本数学原理**

如果某非线性规划的目标函数为自变量的二次函数，约束条件全是线性函数，就称这种规划为二次规划。其数学模型为：

其中，**H**, **A**,和**Aeq**为矩阵,**f**, **b**, **beq**, **lb**, **ub**,和**x**为向量。

**9.2.4.2 相关函数介绍**

quadprog函数

**功能**：求解二次规划问题。

**语法**：

x = quadprog(H,f,A,b)

x = quadprog(H,f,A,b,Aeq,beq)

x = quadprog(H,f,A,b,Aeq,beq,lb,ub)

x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)

x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)

[x,fval] = quadprog(...)

[x,fval,exitflag] = quadprog(...)

[x,fval,exitflag,output] = quadprog(...)

[x,fval,exitflag,output,lambda] = quadprog(...)

**描述**：

x = quadprog(H,f,A,b) 返回向量x，最小化函数 1/2\*x'\*H\*x + f'\*x ，其约束条件为 A\*x <= b。

x = quadprog(H,f,A,b,Aeq,beq)仍然求解上面的问题，但添加了等式约束条件Aeq\*x = beq。

x = quadprog(H,f,A,b,lb,ub)定义设计变量的下界lb和上界ub，使得lb <= x <= ub。

x = quadprog(H,f,A,b,lb,ub,x0)同上，并设置初值x0。

x = quadprog(H,f,A,b,lb,ub,x0,options)根据options参数指定的优化参数进行最小化。

[x,fval] = quadprog(...)返回解x处的目标函数值fval = 0.5\*x'\*H\*x + f'\*x。

[x,fval,exitflag] = quadprog(...)返回exitflag参数，描述计算的退出条件。

[x,fval,exitflag,output] = quadprog(...)返回包含优化信息的结构输出output。

[x,fval,exitflag,output,lambda] = quadprog(...)返回解x处包含拉格朗日乘子的lambda参数。

**变量**：

各变量的意义同前。

**注意**：

1．          一般地，如果问题不是严格凸性的，用quadprog函数得到的可能是局部最优解。

2．          如果用Aeq和Beq明确地指定等式约束，而不是用lb和ub指定，则可以得到更好的数值解。

3．          若x的组分没有上限或下限，则quadprog函数希望将对应的组分设置为Inf（对于上限）或-Inf（对于下限），而不是强制性地给予上限一个很大的数或给予下限一个很小的负数。

4．          对于大型优化问题，若没有提供初值x0，或x0不是严格可行，则quadprog函数会选择一个新的初始可行点。

5．          若为等式约束，且quadprog函数发现负曲度(negative curvature)，则优化过程终止，exitflag的值等于-1。

**算法**：

大型优化算法 当优化问题只有上界和下界，而没有线性不等式或等式约束，则缺省算法为大型算法。或者，如果优化问题中只有线性等式，而没有上界和下界或线性不等式时，缺省算法也是大型算法。

本法是基于内部映射牛顿法(interior-reflective Newton method)的子空间置信域法(subspace trust-region)。该法的具体算法请参见文献[2]。该法的每一次迭代都与用PCG法求解大型线性系统得到的近似解有关。

中型优化算法 quadprog函数使用活动集法，它也是一种投影法，首先通过求解线性规划问题来获得初始可行解。

**诊断**：

1.     大型优化问题 大型优化问题不允许约束上限和下限相等，如若lb(2)==ub(2)，则给出以下出错信息：

Equal upper and lower bounds not permitted in this large-scale method.Use equality constraints and the medium-scale method instead.

若优化模型中只有等式约束，仍然可以使用大型算法；如果模型中既有等式约束又有边界约束，则必须使用中型方法。

2．中型优化问题 当解不可行时，quadprog函数给出以下警告：

Warning: The constraints are overly stringent;there is no feasible solution.

这里，quadprog函数生成使约束矛盾最坏程度最小的结果。

当等式约束不连续时，给出下面的警告信息：

Warning: The equality constraints are overly stringent;there is no feasible solution.

当Hessian矩阵为负半定时，生成无边界解，给出下面的警告信息：

Warning: The solution is unbounded and at infinity;the constraints are not restrictive enough.

这里，quadprog函数返回满足约束条件的x值。

**局限性**：

1．          此时，显示水平只能选择'off'和'final'，迭代参数'iter'不可用。

2．          当问题不定或负定时，常常无解（此时exitflag参数给出一个负值，表示优化过程不收敛）。若正定解存在，则quadprog函数可能只给出局部极小值，因为问题可能时非凸的。

3．          对于大型问题，不能依靠线性等式，因为Aeq必须是行满秩的，即Aeq的行数必须不多于列数。若不满足要求，必须调用中型算法进行计算。

**9.2.4.3 应用实例**

**[例一]** 求解下面的最优化问题：

目标函数

约束条件

首先，目标函数可以写成下面的矩阵形式：

其中

输入下列系数矩阵：

H = [1 -1; -1 2]

f = [-2; -6]

A = [1 1; -1 2; 2 1]

b = [2; 2; 3]

lb = zeros(2,1)

然后调用二次规划函数quadratic：

[x,fval,exitflag,output,lambda] = quadprog(H,f,A,b,[],[],lb)

得问题的解

x =

0.6667

1.3333

fval =

-8.2222

exitflag =

1

output =

iterations: 3

algorithm: 'medium-scale: active-set'

firstorderopt: []

cgiterations: []

lambda.ineqlin

ans =

3.1111

0.4444

0

lambda.lower

ans =

0

0

磁盘中本问题的M文件为opt24\_1.m。

**9.2.5 有约束最小化**

**9.2.5.1 基本数学原理**

在有约束最优化问题中，通常要将该问题转换为更简单的子问题，这些子问题可以求解并作为迭代过程的基础。早期的方法通常是通过构造惩罚函数等来将有约束的最优化问题转换为无约束最优化问题进行求解。现在，这些方法已经被更有效的基于K-T（Kuhn-Tucker）方程解的方法所取代。K-T方程是有约束最优化问题求解的必要条件。假设有所谓的Convex规划问题，f(x)和Gi(x)，i=1,2,…,m为Convex函数，则K-T方程对于求得全局极小点是必要的，也是充分的。

对于规划问题

其中，x是设计参数向量，（x∈Rn），f(x)为目标函数，返回标量值，向量函数G(x)返回等式约束和不等式约束在x处的值。

它的K-T方程可表达为：

（\*）

i=1,…,m

i=me+1,…,m

其中第一行描述了目标函数和约束条件在解处梯度的取消。由于梯度取消，需要用拉格朗日乘子λi(i=1,2,…,m)来平衡目标函数与约束梯度间大小的差异。

K-T方程的解形成了许多非线性规划算法的基础，这些算法直接计算拉格朗日乘子，通过用拟牛顿法更新过程，给K-T方程积累二阶信息，可以保证有约束拟牛顿法的超线性收敛。这些方法称为序列二次规划法（SQP），因为在每一次主要的迭代中都求解一次二次规划问题。

对于给定的规划问题，序列二次规划（SQP）的主要的思路是形成基于拉格朗日函数二次近似的二次规划子问题，即

这里，通过假设约束条件为不等式约束来使（\*）式得到了简化，通过非线性有约束问题线性化来获得二次规划子问题。

二次规划子问题可表达为

i=1,…,me

i=me+1,…,m

该子问题可以用任意一种二次规划算法求解，求得的解可以用来形成新的迭代公式xk+1=xk+αkdk。

用SQP法求解非线性有约束问题时的迭代次数常比用解无约束问题时的少，因为在搜索区域内，SQP方法可以获得最佳的搜索方向和步长信息。

给Rosenbrock函数添加非线性不等式约束g(x)

经过96次迭代得到问题的解：x=[0.9072,0.8288]，初值为x=[-1.9,2]，无约束问题则需要140次迭代。图9-3是搜索路径图。

**图9-3 SQP法的搜索路径**

MATLAB中SQP法的实现分三步，即

●    拉格朗日函数Hessian矩阵的更新；

●    二次规划问题求解；

●    一维搜索和目标函数的计算

(一)、Hessian矩阵的更新

在每一次主要迭代过程中，都用BFGS法计算拉格朗日函数的Hessian矩阵的拟牛顿近似矩阵。更新公式为：

其中：

上式中，λi(i=1,2,…,m)为拉格朗日乘子的估计。

（二）、二次规划求解

SQP法的每一次主要迭代过程中都要求一次二次规划问题，形式如下：

i=1,…,me

i=me+1,…,m

求解过程分两步，第一步涉及可行点（若存在）的计算，第二步为可行点至解的迭代序列。

在第一步中，需要有可行点作为初值，若当前点不可行，则通过求解下列线性规划问题可以得到一个可行点：

i=1,…,me

i=me+1,…,m

其中，Ai为矩阵A的第i行。

（三）、一维搜索和目标函数的计算

二次规划子问题的解生成一个向量dk，它形成一个新的迭代公式：

αk为步长参数。

目标函数的形式如下：

其中，

i=1,…,m

**9.2.5.2 相关函数介绍**

fmincon函数

**功能**：求多变量有约束非线性函数的最小值。

**数学模型**：

其中，**x, b, beq, lb,**和**ub**为向量， **A** 和 **Aeq** 为矩阵， **c(x)** 和 **ceq(x)**为函数，返回标量。**f(x)**, **c(x)**, 和 **ceq(x)**可以是非线性函数。

**语法**：

x = fmincon(fun,x0,A,b)

x = fmincon(fun,x0,A,b,Aeq,beq)

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub)

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2, ...)

[x,fval] = fmincon(...)

[x,fval,exitflag] = fmincon(...)

[x,fval,exitflag,output] = fmincon(...)

[x,fval,exitflag,output,lambda] = fmincon(...)

[x,fval,exitflag,output,lambda,grad] = fmincon(...)

[x,fval,exitflag,output,lambda,grad,hessian] = fmincon(...)

**描述**：

fmincon 求多变量有约束非线性函数的最小值。该函数常用于有约束非线性优化问题。

x = fmincon(fun,x0,A,b) 给定初值x0，求解fun函数的最小值x。fun函数的约束条件为A\*x <= b，x0可以是标量、向量或矩阵。

x = fmincon(fun,x0,A,b,Aeq,beq) 最小化fun函数，约束条件为Aeq\*x = beq 和 A\*x <= b。若没有不等式存在，则设置A=[]、b=[]。

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub) 定义设计变量x的下界lb和上界ub，使得总是有lb <= x <= ub。若无等式存在，则令Aeq=[]、beq=[]。

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon) 在上面的基础上，在nonlcon参数中提供非线性不等式c(x)或等式ceq(x)。 fmincon函数要求c(x) <= 0且ceq(x) = 0。当无边界存在时，令lb=[]和（或）ub=[]。

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options) 用optiions参数指定的参数进行最小化。

x = fmincon(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2,...) 将问题参数P1, P2等直接传递给函数fun和nonlin。若不需要这些变量，则传递空矩阵到A, b, Aeq, beq, lb, ub, nonlcon和 options。

[x,fval] = fmincon(...) 返回解x处的目标函数值。

[x,fval,exitflag] = fmincon(...) 返回exitflag参数，描述函数计算的退出条件。

[x,fval,exitflag,output] = fmincon(...) 返回包含优化信息的输出参数output。

[x,fval,exitflag,output,lambda] = fmincon(...) 返回解x处包含拉格朗日乘子的lambda参数。

[x,fval,exitflag,output,lambda,grad] = fmincon(...) 返回解x处fun函数的梯度。

[x,fval,exitflag,output,lambda,grad,hessian] = fmincon(...) 返回解x处fun函数的Hessian矩阵。

**变量**：

nonlcon参数

该参数计算非线性不等式约束c(x)<=0 和非线性等式约束ceq(x)=0。 nonlcon 参数是一个包含函数名的字符串。该函数可以是M文件、内部文件或MEX文件。它要求输入一个向量x，返回两个变量—解x处的非线性不等式向量c和非线性等式向量ceq。例如，若nonlcon='mycon'，则M文件mycon.m具有下面的形式：

function [c,ceq] = mycon(x)

c = ... % 计算x处的非线性不等式。

ceq = ... % 计算x处的非线性等式。

若还计算了约束的梯度，即

options = optimset('GradConstr','on')

则nonlcon函数必须在第三个和第四个输出变量中返回c(x)的梯度GC和ceq(x)的梯度Gceq。当被调用的nonlcon函数只需要两个输出变量（此时优化算法只需要c和ceq的值，而不需要GC和GCeq）时，可以通过查看nargout的值来避免计算GC和Gceq的值。

function [c,ceq,GC,GCeq] = mycon(x)

c = ... % 解x处的非线性不等式。

ceq = ... % 解x处的非线性等式。

if nargout > 2 % 被调用的nonlcon函数，要求有4个输出变量。

GC = ... % 不等式的梯度。

GCeq = ... % 等式的梯度。

end

若nonlcon函数返回m元素的向量c和长度为n的x，则c(x)的梯度GC是一个n\*m的矩阵，其中GC(i,j)是c(j)对x(i)的偏导数。同样，若ceq是一个p元素的向量，则ceq(x)的梯度Gceq是一个n\*p的矩阵，其中Gceq(i,j)是ceq(j)对x(i)的偏导数。

其它参数意义同前。

**注意**：

大型优化问题：

1． 使用大型算法，必须在fun函数中提供梯度信息(options.GradObj设置为'on')。如果没有梯度信息，则给出警告信息。

fmincon函数允许g(x)为一近似梯度，但使用真正的梯度将使优化过程更具稳健性。

2．当对矩阵的二阶导数（即Hessian矩阵）进行计算以后，用该函数求解大型问题将更有效。但不需要求得真正的Hessian矩阵，如果能提供Hessian矩阵的稀疏结构的信息（用options参数的HessPattern属性），则fmincon函数可以算得Hessian矩阵的稀疏有限差分近似。

3．若x0不是严格可行的，则fmincon函数选择一个新的严格可行初始点。

4．若x的某些元素没有上界或下界，则fmincon函数更希望对应的元素设置为Inf（对于上界）或-Inf（对于下界），而不希望强制性地给上界赋一个很大的值，给下界赋一个很小的负值。

5．线性约束最小化课题中也有几个问题需要注意：

  Aeq矩阵中若存在密集列或近密集列（A dense (或fairly dense)column），会导致满秩并使计算费时。

  fmincon函数剔除Aeq中线性相关的行。此过程需要进行反复的因子分解，因此，如果相关行很多的话，计算将是一件很费时的事情。

    
每一次迭代都要用下式进行稀疏最小二乘求解

其中**RT**为前提条件的乔累斯基因子。

中型优化问题

1．如果用Aeq和beq清楚地提供等式约束，将比用lb和ub获得更好的数值解。

2．在二次子问题中，若有等式约束并且因等式（dependent equalities）被发现和剔除的话，将在过程标题中显示'dependent'(当output参数要求使用options.Display = 'iter')。只有在等式连续的情况下，因等式才会被剔除。若等式系统不连续，则子问题将不可行并在过程标题中打印'infeasible'信息。

**算法**：

大型优化算法 缺省时，若提供了函数的梯度信息，并且只有上下界存在或只有线性等式约束存在，则fmincon函数将选择大型算法。本法是基于内部映射牛顿法(interior-reflective Newton method)的子空间置信域法(subspace trust-region)。该法的具体算法请参见文献[2]。该法的每一次迭代都与用PCG法求解大型线性系统得到的近似解有关。

中型优化算法 fmincon函数使用序列二次规划法(SQP)。本法中，在每一步迭代中求解二次规划子问题，并用BFGS法更新拉格朗日Hessian矩阵。参见文献[3]、[6]。

该算法使用与文献[1]、[2]和[3]中提供的相似的目标函数进行一维搜索。二次子问题使用文献[4]描述的活动集方案进行求解。

**诊断**：

大型优化问题 求大型优化问题的代码中不允许上限和下限相等，即不能有lb(2)==ub(2),否则给出下面的出错信息：

Equal upper and lower bounds not permitted in this large-scale

method.

Use equality constraints and the medium-scale method instead.

若只有等式约束，可以仍然使用大型算法。当既有等式约束又有边界约束时，使用中型算法。

**局限**：

1．          目标函数和约束函数都必须是连续的，否则可能会给出局部最优解。

2．          当问题不可行时，fmincon函数将试图使最大约束值最小化。

3．          目标函数和约束函数都必须是实数。

4．          对于大型优化问题，使用大型优化算法时，用户必须在fun函数中提供梯度(options参数的GradObj属性必须设置为'on') , 并且只可以指定上界和下界约束，或者只有线性约束必须存在，Aeq的行数不能多于列数。

5．          现在，如果在fun函数中提供了解析梯度，选项参数DerivativeCheck不能与大型方法一起用，以比较解析梯度和有限差分梯度。可以通过将options参数的MaxIter属性设置为0来用中型方法核对导数。然后用大型方法求解问题。

**9.2.5.3 应用实例**

**[例一]** 求解下列优化问题

目标函数

约束条件

首先编写一个M文件，返回x处的函数值opt25\_1o.m。

function f = myfun(x)

f = -x(1) \* x(2) \* x(3);

然后将约束条件改写成下面形式的不等式

因为两个约束条件都是线性的，将它们表达为矩阵不等式的形式，其中，

下一步给定初值，并调用优化过程

x0 = [10; 10; 10]; % 初值

A=[-1 –2 –2

1 2 2];

b=[0;72];

[x,fval] = fmincon(@opt25\_1o,x0,A,b)

经过66次函数评价以后，得到问题的解

x =

24.0000

12.0000

12.0000

函数值为

fval =

-3.4560e+03

而且线性不等式约束的值<= 0

A\*x-b=

-72

0

磁盘中本问题的M文件为opt25\_1.m。

**[例二]** 求侧面积为常数6\*52m2的体积最大的长方体体积。

设该长方体的长、宽、高分别为**x1**、**x2**和**x3**，据题意得到下面的数学模型：

编一个M文件optim25\_2o.m，返回x处的函数值f。

function f = myfun(x)

f = -x(1) \* x(2) \* x(3);

由于约束条件是非线性等式约束，所以需要编写一个约束条件M文件opt25\_2c.m：

function [c,ceq] = mycon(x)

ceq =x(2)x(3)+x(3)x(1)+x(1)x(2)-75

下一步给定初值，并调用优化过程(磁盘中M文件名为opt25\_2.m)。

x0 = [4; 5; 6];

lb=zeros(3,1);

[x,fval,exitflag,output,lambda]=fmincon(@opt25\_2o,x0,[],[],[],[], lb,[],@opt25\_2c)

计算结果为：

Optimization terminated successfully:

Search direction less than 2\*options.TolX and

maximum constraint violation is less than options.TolCon

Active Constraints:

1

x =

5.0000

5.0000

5.0000

fval =

-125.0000

exitflag =

1

output =

iterations: 7

funcCount: 38

stepsize: 1

algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'

firstorderopt: []

cgiterations: []

lambda =

lower: [3x1 double]

upper: [3x1 double]

eqlin: [0x1 double]

eqnonlin: 2.5000

ineqlin: [0x1 double]

ineqnonlin: [0x1 double]

优化结果显示过程成功收敛，搜索方向小于两倍options.TolX，最大违约值小于options.TolCon，主动约束为1个。

问题的解为x(1)=x(2)=x(3)=5.0000m，最大体积为125.0000m3。exitflag=1，表示过程成功收敛于解x处。output输出变量显示了收敛过程中的迭代次数、目标函数计算次数、步长、算法等信息。lambda则包含模型信息。

**[例三]** 试设计一压缩圆柱螺旋弹簧，要求其质量最小。弹簧材料为65Mn，最大工作载荷Pmax=40N，最小工作载荷为0，载荷变化频率fr=25Hz，弹簧寿命为104h，弹簧钢丝直径d的取值范围为1-4mm，中径D2的取值范围为10-30mm，工作圈数n不应小于4.5圈，弹簧旋绕比C不应小于4，弹簧一端固定，一端自由，工作温度为50C，弹簧变形量不小于10mm。

本题的优化目标是使弹簧质量最小，圆柱螺旋弹簧的质量可以表示为

式中， --弹簧材料的密度，对于钢材 =7.8×10-6kg/mm3；

n—工作圈数；

n2—死圈数，常取n2=1.5-2.5，现取n2=2；

D2—弹簧中径，mm；

d—弹簧钢丝直径，mm。

将已知参数代入公式，进行整理以后得到问题的目标函数为

根据弹簧性能和结构上的要求，可写出问题的约束条件：

1． 强度条件

2． 刚度条件

3． 稳定性条件

4． 不发生共振现象，要求

5． 弹簧旋绕比的限制

6． 对d,n,D2的限制

且应取标准值，即1.0,1.2,1.6,2.0,2.5,3.0,3.5,4.0mm等。

由上可知，该压缩圆柱螺旋弹簧的优化设计是一个三维的约束优化问题，其数学模型为：

取初始设计参数为**X(0)=[2.0,5.0,25.0]T**

首先编写目标函数的M文件opt25\_3.m，返回x处的函数值f。

function f = myfun(x)

f = 0.192457\*1e-4\*( x(2)+2)\*x(1)^2 \* x(3);

由于约束条件中有非线性约束，所以需要编写一个描述非线性约束条件的M文件opt25\_3c.m：

function [c,ceq] = mycon(x)

c(1)=350-163\*x(1)^(-2.86)\*x(3)^0.86;

c(2)=10-0.4\*0.01\*x(1)^(-4)\*x(2)\*x(3)^3;

c(3)=(x(2)+1.5)\*x(1)+0.44\*0.01\*x(1)^(-4)\*x(2)\*x(3)^3-3.7\*x(3);

c(4)=375-0.356\*1e6\*x(1)\*x(2)^(-1)\*x(3)^(-2);

c(5)=4-x(3)/x(1);

然后设置线性约束的系数：

A=[-1 0 0

1 0 0

0 –1 0

0 1 0

0 0 –1

0 0 1];

b=[-1;4;-4.5;50;-10;30];

下一步给定初值，给定变量的下限约束，并调用优化过程(磁盘中M文件为opt25\_3.m)

x0 = [2.0; 5.0; 25.0];

lb=zeros(3,1);

[x,fval,exitflag,output,lambda]=fmincon(@opt25\_3o,x0,A,b,[],[], lb,[],@opt25\_3c)

计算结果为：

x =

1.6564

4.5000

16.1141

fval =

0.0055

exitflag =

1

output =

iterations: 3

funcCount: 16

stepsize: 1

algorithm: 'medium-scale: SQP, Quasi-Newton, line-search'

firstorderopt: []

cgiterations: []

所以当弹簧钢丝的直径d、工作圈数n及中径D2分别取1.6564、4.5000和16.1141时弹簧质量最小，为5.5克。考虑到实际情况，各参数可分别取1.6、5.0和16.0

x=[1.6 5.0 16.0];

z=optim253(x)

z =

0.0055

故此时弹簧质量仍为5.5克。

**9.2.6 目标规划问题**

**9.2.6.1 基本数学原理**

前面介绍的最优化方法只有一个目标函数，是单目标最优化方法。但是，在许多实际工程问题中，往往希望多个指标都达到最优值，所以它有多个目标函数。这种问题称为多目标最优化问题。

多目标最优化问题的数学模型为：

i=1,…,me

i=me+1,…,m

其中F(x)为目标函数向量。

由于多目标最优化问题中各目标函数之间往往是不可公度的，因此往往没有唯一解，此时必须引进非劣解的概念（非劣解又称为有效解或帕累托解）。

定义：若x\*(x\*∈Ω)的邻域内不存在Δx，使得（x\*+Δx）∈Ω，且

i=1,…,m

对于某些j

则称为x\*非劣解。

多目标规划有许多解法，下面列出常用的几种：

1.   权和法

该法将多目标向量问题转化为所有目标的加权求和的标量问题，即

加权因子的选取方法很多，有专家打分法、α方法、容限法和加权因子分解法等。

该问题可以用标准的无约束最优化算法进行求解。

2.   ε约束法

ε约束法克服了权和法的某些凸性问题。它对目标函数向量中的主要目标Fp进行最小化，将其它目标用不等式约束的形式写出：

sub. i=1,…,m i≠p

3.   目标达到法

目标函数系列为F(x)={F1(x), F2(x),…, Fm(x)}，对应地有其目标值系列 。允许目标函数有正负偏差，偏差的大小由加权系数向量W={W1,W2,…,Wm}控制，于是目标达到问题可以表达为标准的最优化问题:

sub. i=1,…,m

指定目标{ }，定义目标点P。权重向量定义从P到可行域空间Λ（γ）的搜索方向，在优化过程中，γ的变化改变可行域的大小，约束边界变为唯一解点F1s、F2s。

4.目标达到法的改进

目标达到法的一个好处是可以将多目标最优化问题转化为非线性规划问题，但是，在序列二次规划（SQP）过程中，一维搜索的目标函数选择不是一件容易的事情，因为在很多情况下，很难决定是使目标函数变大好还是使它变小好。这导致许多目标函数创建过程的提出。可以通过将目标达到问题变为最大最小化问题来获得更合适的目标函数。

其中

i=1,…,m

**9.2.6.2 相关函数介绍**

fgoalattain函数

**功能**：求解多目标达到问题

**数学模型**：

其中**x**, **weight**, **goal**, **b**, **beq**, **lb**和**ub** 为向量, **A**和**Aeq**为矩阵, **c(x)**, **ceq(x)**和**F(x)**为函数，返回向量。**F(x), c(x)**和**ceq(x)**可以是非线性函数。

**语法**：

x = fgoalattain(fun,x0,goal,weight)

x = fgoalattain(fun,x0,goal,weight,A,b)

x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq)

x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub)

x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon)

x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,...

lb,ub,nonlcon,options)

x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,...

lb,ub,nonlcon,options,P1,P2,...)

[x,fval] = fgoalattain(...)

[x,fval,attainfactor] = fgoalattain(...)

[x,fval,attainfactor,exitflag] = fgoalattain(...)

[x,fval,attainfactor,exitflag,output] = fgoalattain(...)

[x,fval,attainfactor,exitflag,output,lambda] = fgoalattain(...)

**描述**：

fgoalattain求解多目标达到问题。

x = fgoalattain(fun,x0,goal,weight)试图通过变化x来使目标函数fun达到goal指定的目标。初值为x0,weight参数指定权重。

x = fgoalattain(fun,x0,goal,weight,A,b)求解目标达到问题，约束条件为线性不等式A\*x <= b。

x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq)求解目标达到问题，除提供上面的线性不等式以外，还提供线性等式Aeq\*x = beq 。当没有不等式存在时，设置A=[]、b=[]。

x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub)为设计变量x定义下界lb和上界ub集合，这样始终有lb <= x <= ub。

x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon) 将目标达到问题归结为nonlcon参数定义的非线性不等式c(x)或非线性等式ceq(x)。fgoalattain函数优化的约束条件为(x) <= 0和ceq(x) = 0。若不存在边界，设置lb=[]和（或）ub=[]。

x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon,... options)用options中设置的优化参数进行最小化。

x = fgoalattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon,... options,P1,P2,...) 将问题参数P1, P2等直接传递给函数fun和nonlcon。如果不需要参数A, b, Aeq, beq, lb, ub, nonlcon和options，将它们设置为空矩阵。

[x,fval] = fgoalattain(...) 返回解x处的目标函数值。

[x,fval,attainfactor] = fgoalattain(...) 返回解x处的目标达到因子。

[x,fval,attainfactor,exitflag] = fgoalattain(...)返回exitflag参数，描述计算的退出条件。

[x,fval,attainfactor,exitflag,output] = fgoalattain(...)返回包含优化信息的输出参数output。

[x,fval,attainfactor,exitflag,output,lambda] = fgoalattain(...) 返回包含拉格朗日乘子的lambda参数。

**变量**：

goal变量

目标希望达到的向量值。向量的长度与fun函数返回的目标数F相等。fgoalattain函数试图通过最小化向量F中的值来达到goal参数给定的目标。

nonlcon参数：

该函数计算非线性不等式约束c(x) <=0和非线性等式约束ceq(x)=0。nonlcon函数是一个包含函数名的字符串，该函数可以是M文件、内部函数或MEX文件。nonlcon函数需要输入向量x，返回两个变量—x处的非线性不等式向量c和x处的非线性等式向量ceq。例如，若nonlcon='mycon'，则M文件的形式如下：

function [c,ceq] = mycon(x)

c = ... % 计算x处的非线性不等式。

ceq = ... % 计算x处的非线性等式。

若约束函数的梯度可以计算，且options.GradConstr设为'on',即

options = optimset('GradConstr','on')

则函数nonlcon也必须在第三个和第四个输出变量中输出c(x)的梯度GC和ceq(x)的梯度GCeq。注意，可以通过核对nargout参数来避免计算GC和GCeq。

function [c,ceq,GC,GCeq] = mycon(x)

c = ... % x处的非线性不等式。

ceq = ... % x处的非线性等式。

if nargout > 2 % 被调用的nonlcon函数，有4个输出。

GC = ... % 不等式的梯度。

GCeq = ... % 等式的梯度。

end

若nonlcon函数返回m元素的向量c和长度为n的x，则c(x)的梯度GC是一个n\*m的矩阵，其中GC(i,j)是c(j)对x(i)的偏导数。同样，若ceq是一个p元素的向量，则ceq(x)的梯度Gceq是一个n\*p的矩阵，其中Gceq(i,j)是ceq(j)对x(i)的偏导数。

options变量

优化参数选项。你可以用optimset函数设置或改变这些参数的值。

DerivativeCheck – 比较用户提供的导数（目标函数或约束函数的梯度）和有限差分导数。

Diagnostics – 打印将要最小化或求解的函数的诊断信息。

DiffMaxChange – 变量中有限差分梯度的最大变化。

DiffMinChange - 变量中有限差分梯度的最小变化。

Display – 显示水平。设置为'off'时不显示输出；设置为'iter'时显示每一次迭代的输出；设置为'final'时只显示最终结果。

GoalExactAchieve – 使得目标个数刚好达到，不多也不少。

GradConstr – 用户定义的约束函数的梯度。

GradObj – 用户定义的目标函数的梯度。使用大型方法时必须使用梯度，对于中型方法则是可选项。

MaxFunEvals – 函数评价的允许最大次数。

MaxIter – 函数迭代的允许最大次数。

MeritFunction – 如果设为'multiobj'，则使用目标达到或最大最小化目标函数的方法。若设置为'singleobj'，则使用fmincon 函数计算目标函数。

TolCon – 约束矛盾的终止容限。

TolFun – 函数值处的终止容限。

TolX – x处的终止容限。

weight变量

为权重向量，可以控制低于或超过fgoalattain函数指定目标的相对程度。当goal的值都是非零值时，为了保证活动对象超过或低于的比例相当，将权重函数设置为abs(goal) (活动对象为阻止解处目标改善的对象集合)。

注意:

1.              当目标值中的任意一个为零时，设置weight=abs(goal)将导致目标约束看起来更象硬约束，而不象目标约束。

2.              当加权函数weight为正时，fgoalattain函数试图使对象小于目标值。为了使目标函数大于目标值，将权重weight设置为负。为了使目标函数尽可能地接近目标值，使用GoalsExactAchieve参数，将fun函数返回的第一个元素作为目标。

attainfactor变量

attainfactor变量是超过或低于目标的个数。若attainfactor为负，则目标已经溢出；若attainfactor为正，则目标个数还未达到。

其它参数意义同前。

**注意**：

当特征值为复数时，本问题不连续，这也说明了为什么收敛速度很慢。尽管原始方法假设函数是连续的，该法仍然可以向解的方向前进，因为在解的位置上，没有发生不连续的现象。当对象和目标为复数时，fgoalattain函数将试图得到最小二乘意义上的目标。

**算法**：

多目标优化同时涉及到一系列对象。fgoalattain函数求解该问题的基本算法是目标达到法。该法为目标函数建立起目标值。多目标优化的具体算法在前面进行了更详细的介绍。

本次实现过程中，使用了松弛变量γ作为模糊变量同时最小化目标向量F(x)；goal参数是一系列目标达到值。在进行优化之前，通常不知道对象是否会达到目标。使用权向量weight可以控制是没有达到还是溢出。

fgoalattain函数使用序列二次规划法(SQP)，前面已经进行了比较多的介绍。算法中对于一维搜索和Hessian矩阵进行了修改。在一维搜索中，将精确目标函数和文献[2]、[3]中的目标函数一起使用。当有一个目标函数不再发生改善时，一维搜索终止。修改的Hessian矩阵借助于本问题的结构，也被采用。详细内容可参见文献[5]、[6]。

attainfactor参数包含解处的γ值。γ取负值时表示目标溢出。

**局限性**：

目标函数必须是连续的。fgoalattain函数将只给出局部最优解。

**参见**：

fmincon, fminimax, optimset

**9.2.6.3 应用实例**

**[例二]** 某化工厂拟生产两种新产品A和B，其生产设备费用分别为：A，2万元/吨；B，5万元/吨。这两种产品均将造成环境污染，设由公害所造成的损失可折算为：A，4万元/吨；B，1万元/吨。由于条件限制，工厂生产产品A和B的最大生产能力各为每月5吨和6吨，而市场需要这两种产品的总量每月不少于7吨。试问工厂如何安排生产计划，在满足市场需要的前提下，使设备投资和公害损失均达最小。该工厂决策认为，这两个目标中环境污染应优先考虑，设备投资的目标值为20万元，公害损失的目标为12万元。

设工厂每月生产产品A为x1吨，B为x2吨，设备投资费为f1(x)，公害损失费为f2(x)，则这个问题可表达为多目标优化问题：

首先需要编写目标函数的M文件opt26\_2o.m，返回目标计算值

function f=myfun(x)

f(1)=2\*x(1)+5\*x(2);

f(2)=4\*x(1)+x(2);

给定目标，权重按目标比例确定，给出初值

goal=[20 12];

weight=[20 12];

x0=[2 5];

给出约束条件的系数

A=[1 0;0 1;-1 -1];

b=[5 6 -7];

lb=zeros(2,1);

[x,fval,attainfactor,exitflag] = …

fgoalattain(@opt26\_2o,x0,goal,weight,A,b,[],[],lb,[])

磁盘中上面程序段的M文件为opt26\_2.m。

计算结果为

x =

2.9167 4.0833

fval =

26.2500 15.7500

attainfactor =

0.3125

exitflag =

1

故工厂每月生产产品A为2.9167吨，B为4.0833吨。设备投资费和公害损失费的目标值分别为26.2500万元和15.7500万元。达到因子为0.3125，计算收敛。

**[例三]** 某厂生产两种产品A和B，已知生产A产品100kg需8个工时，生产B产品100kg需10个工时。假定每日可用的工时数为40，且希望不雇临时工，也不加班生产。这两种产品每100kg均可获利100元。此外，有个顾客要求每日供应他B种产品600kg。问应如何安排生产计划？

设生产A、B两种产品的数量分别为x1和x2(均以100kg计)，为了使生产计划比较合理，要求用人尽量少，获利尽可能多，另外B种产品的产量尽量多。由题意建立下面的数学模型：

首先需要编写目标函数的M文件opt26\_3o.m，返回目标计算值

function f=myfun(x)

f(1)=8\*x(1)+10\*x(2);

f(2)=-100\*x(1)-100\*x(2);

f(3)=-x(2);

给定目标，权重按目标比例确定，给出初值

goal=[40 –800 -6];

weight=[40 –800 -6];

x0=[2 2];

给出约束条件的系数

A=[8 10;0 -1];

b=[40 -6];

lb=zeros(2,1);

options=optimset('MaxFunEvals',5000); % 设置函数评价的最大次数为5000次

[x,fval,attainfactor,exitflag] = …

fgoalattain(@opt26\_3o,x0,goal,weight,A,b,[],[],lb,[],[]，options);

磁盘中上面程序段的M文件为opt26\_3.m。

计算结果为

x =

2.0429 1.9458

fval =

35.8007 -398.8648 -1.9458

attainfactor =

-0.0646

exitflag =

0

经过5000次迭代以后，生产A、B两种产品的数量各为204.29kg和194.58kg。

**[例四]** 某工厂因生产需要欲采购一种原材料，市场上的这种原料有两个等级，甲级单价2元/千克，乙级单价1元/千克。要求所花总费用不超过200元，购得原料总量不少于100千克，其中甲级原料不少于50千克，问如何确定最好的采购方案。

设x1、x2分别为采购甲级和乙级原料的数量（千克），要求采购总费用尽量少，采购总重量尽量多，采购甲级原料尽量多。由题意可得：

首先需要编写目标函数的M文件opt26\_4o.m，返回目标计算值

function f=myfun(x)

f(1)=2\*x(1)+ x(2);

f(2)=-x(1)- x(2);

f(3)=-x(1);

给定目标，权重按目标比例确定，给出初值

goal=[200 –100 -50];

weight=[2040 –100 -50];

x0=[55 55];

给出约束条件的系数

A=[2 1;-1 –1;-1 0];

b=[200 -100 -50];

lb=zeros(2,1);

[x,fval,attainfactor,exitflag] = …

fgoalattain(@opt26\_4o,x0,goal,weight,A,b,[],[],lb,[]);

磁盘中本问题的M文件为opt26\_4.m。

输出计算结果

x =

50 50

fval =

150 -100 -50

attainfactor =

1.3235e-023

exitflag =

1

所以，最好的采购方案是采购甲级原料和乙级原料各50千克。此时采购总费用为150元，总重量为100千克，甲级原料总重量为50千克。

**9.2.7 最大最小化**

**9.2.7.1 基本数学原理**

通常我们遇到的都是目标函数的最大化和最小化问题，但是在某些情况下，则要求最大值的最小化才有意义。例如城市规划中需要确定急救中心、消防中心的位置，可取的目标函数应该是到所有地点最大距离的最小值，而不是到所有目的地的距离和为最小。这是两种完全不同的准则，在控制理论、逼近论、决策论中也使用最大最小化原则。

最大最小化问题的数学模型为

其中**x, b, beq, lb**和**ub**为向量，**A**和**Aeq**为矩阵,**c(x)**, **ceq(x)**和**F(x)**为函数，返回向量。**F(x), c(x)**和**ceq(x)**可以是非线性函数。

Matlab优化工具箱中采用序列二次规划法求解最大最小化问题。

**9.2.7.2 相关函数介绍**

fminimax函数

**功能**：求解最大最小化问题。

**语法**：

x = fminimax(fun,x0)

x = fminimax(fun,x0,A,b)

x = fminimax(fun,x0,A,b,Aeq,beq)

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub)

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon)

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options)

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2,...)

[x,fval] = fminimax(...)

[x,fval,maxfval] = fminimax(...)

[x,fval,maxfval,exitflag] = fminimax(...)

[x,fval,maxfval,exitflag,output] = fminimax(...)

[x,fval,maxfval,exitflag,output,lambda] = fminimax(...)

**描述**：

fminimax 使多目标函数中的最坏情况达到最小化。给定初值估计，该值必须服从一定的约束条件。

x = fminimax(fun,x0)初值为x0，找到fun函数的最大最小化解x。

x = fminimax(fun,x0,A,b)给定线性不等式A\*x <= b，求解最大最小化问题。

x = fminimax(fun,x,A,b,Aeq,beq) 给定线性等式，Aeq\*x = beq，求解最大最小化问题。如果没有不等式存在，设置A=[]、b=[]。

x = fminimax(fun,x,A,b,Aeq,beq,lb,ub) 为设计变量定义一系列下限lb和上限ub，使得总有lb <= x <= ub。

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon) 在nonlcon参数中给定非线性不等式约束c(x)或等式约束ceq(x)，fminimax函数要求c(x) <= 0且ceq(x) = 0。若没有边界存在，设置lb=[]和（或）ub=[]。

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options) 用options给定的参数进行优化。

x = fminimax(fun,x0,A,b,Aeq,beq,lb,ub,nonlcon,options,P1,P2,...)将问题参数P1, P2等直接传递给函数fun和nonlcon。如果不需要变量A, b, Aeq, beq, lb, ub, nonlcon和options将它们设置为空矩阵。

[x,fval] = fminimax(...) 返回解x处的目标函数值。

[x,fval,maxfval] = fminimax(...)返回解x处的最大函数值。

[x,fval,maxfval,exitflag] = fminimax(...) 返回exitflag参数，描述函数计算的退出条件。

[x,fval,maxfval,exitflag,output] = fminimax(...) 返回描述优化信息的结构输出output参数。

[x,fval,maxfval,exitflag,output,lambda] = fminimax(...) 返回包含解x处拉格朗日乘子的lambda参数。

**变量**：

maxfval变量

解x处函数值的最大值，即，maxfval = max{fun(x)}。

**注意**：

1．          在options.MinAbsMax中设置F最坏绝对值最小化了的目标数。该目标应该放到F的第一个元素中去。例如，考虑上面的问题，需要找到x值，

使得下式的最大绝对值最小化：

通过调用fminimax函数来进行求解

x0 = [0.1; 0.1]; % 设置初值

options = optimset('MinAbsMax',5); % 最小化绝对值

[x,fval] = fminimax(fun,x0,[],[],[],[],[],[],[],options);

经过7次迭代以后，得到问题的解

x =

4.9256

2.0796

fval =

37.2356 -37.2356 -6.8357 -7.0052 -0.9948

2．当提供了等式约束并且在二次子问题中发现并剔除了因等式，则在过程标题中打印'dependent'字样(当输出选项设置为options.Display='iter')。因等式只有在等式连续的情况下才被剔除。若系统不连续，则子问题不可行并且在过程标题中打印'infeasible'字样。

**算法**：

fminimax函数使用序列二次规划法(SQP)进行计算。对一维搜索法和Hessian矩阵的计算进行了修改。在一维搜索中，将精确目标函数和文献[2]、[3]中的目标函数一起使用。当有一个目标函数不再发生改善时，一维搜索终止。修改的Hessian矩阵借助于本问题的结构，也被采用。详细内容可参见文献[5]、[6]。

**局限性**：

目标函数必须连续，否则fminimax函数有可能给出局部最优解。

**参见：**

optimset, fgoalattain, lsqnonlin

**文献**：

[1] Han, S.P., "A Globally Convergent Method For Nonlinear Programming," J. of Optimization Theory and Applications, Vol. 22, p. 297, 1977.

[2] Powell, M.J.D., "A Fast Algorithm for Nonlineary Constrained Optimization Calculations," Numerical Analysis, ed. G.A. Watson, Lecture Notes in Mathematics, Springer Verlag, Vol. 630, 1978.

[3] Brayton, R.K., S.W. Director, G.D. Hachtel, and L.Vidigal, "A New Algorithm for Statistical Circuit Design Based on Quasi-Newton Methods and Function Splitting," IEEE Trans. Circuits and Systems, Vol. CAS-26, pp. 784-794, Sept. 1979.

[4] Grace, A.C.W., "Computer-Aided Control System Design Using Optimization Techniques," Ph.D. Thesis, University of Wales, Bangor, Gwynedd, UK, 1989.

**9.2.7.3 应用实例**

**[例一]** 求解下列最优化问题，使下面各目标函数中的最大值最小：

其中

首先编写一个计算x处五个函数的M文件opt27\_1o.m。

function f = myfun(x)

f(1)= 2\*x(1)^2+x(2)^2-48\*x(1)-40\*x(2)+304;

f(2)= -x(1)^2 - 3\*x(2)^2;

f(3)= x(1) + 3\*x(2) -18;

f(4)= -x(1)- x(2);

f(5)= x(1) + x(2) - 8;

然后调用优化过程(磁盘中M文件为opt27\_1.m)：

x0 = [0.1; 0.1]; % 提供解的初值

[x,fval] = fminimax(@opt27\_1o,x0)

经过7次迭代以后，得到问题的解

x =

4.0000

4.0000

fval =

0.0000 -64.0000 -2.0000 -8.0000 -0.0000

**[例二]** 定位问题

设某城市有某种物品的10个需求点，第i个需求点Pi的坐标为（ai,bi）,道路网与坐标轴平行，彼此正交。现打算建一个该物品的供应中心，且由于受到城市某些条件的限制，该供应中心只能设在x界于[5，8]，y界于[5，8]的范围内。问该中心应建在何处为好？

Pi点的坐标为：

ai: 1 4 3 5 9 12 6 20 17 8

bi: 2 10 8 18 1 4 5 10 8 9

设供应中心的位置为（x,y），要求它到最远需求点的距离尽可能小，由于此处应采用沿道路行走的距离，可知用户Pi到该中心的距离为|x-ai|+|y-bi|，从而可得目标函数如下

约束条件为

首先编写一个计算x处10个目标函数的M文件opt27\_2o.m。

function f = myfun(x)

%输入各个点的坐标值

a=[1 4 3 5 9 12 6 20 17 8];

b=[2 10 8 18 1 4 5 10 8 9];

f(1) = abs(x(1)-a(1))+abs(x(2)-b(1));

f(2) = abs(x(1)-a(2))+abs(x(2)-b(2));

f(3) = abs(x(1)-a(3))+abs(x(2)-b(3));

f(4) = abs(x(1)-a(4))+abs(x(2)-b(4));

f(5) = abs(x(1)-a(5))+abs(x(2)-b(5));

f(6) = abs(x(1)-a(6))+abs(x(2)-b(6));

f(7) = abs(x(1)-a(7))+abs(x(2)-b(7));

f(8) = abs(x(1)-a(8))+abs(x(2)-b(8));

f(9) = abs(x(1)-a(9))+abs(x(2)-b(9));

f(10) = abs(x(1)-a(10))+abs(x(2)-b(10));

然后输入初值、约束条件并调用优化过程进行计算(M文件为opt27\_2.m)：

x0 = [6; 6]; % 提供解的初值

AA=[-1 0

1 0

0 –1

0 1];

bb=[-5;8;-5;8];

[x,fval] = fminimax(@opt27\_2o,x0,AA,bb)

计算结果为：

x =

8

8

fval =

13 6 5 13 8 8 5 14 9 1

可见，在限制区域内的东北角设置供应中心可以使该点到各需求点的最大距离最小。最小最大距离为14个距离单位。